LOAD BALANCING IN CLOUD COMPUTING USING CLOUD SIMULATOR

A PROJECT REPORT

Submitted by

R. BHUVANESHWARI (715517104018)

M.VARSHINI (715517104058)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH,

COIMBATORE-641062.

ANNA UNIVERSITY: CHENNAI 600 025

JUNE 2021

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"LOAD BALANCING IN CLOUD COMPUTING USING CLOUD SIMULATOR"** is the bonafide work of **"R. BHUVANESHWARI (715517104018), M. VARSHINI (715517104058)"** who carried work under my supervision.

SIGNATURE

Dr. R. Manimegalai

HEAD OF THE DEPARTMENT

Computer Science and Engineering

PSG Institute of Technology and

Applied Research,

Coimbatore-641 062.

SIGNATURE

Ms. P. Priya Ponnusamy

SUPERVISOR

Assistant Professor

(Selection Grade)

Computer Science and Engineering

PSG Institute of Technology and

Applied Research,

Coimbatore- 641062.

Submitted for the project viva-voce Examination held on _____.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our deep sense of gratitude to our Managing Trustee **Shri L. Gopalakrishnan** for his invaluable guidance and blessing in the project.

We are very grateful to **Dr. G. Chandramohan, Principal** and professor for providing us with an environment to complete our project successfully.

We also take privilege for expressing my gratitude to **Dr. P.V. Mohanram**, **Secretary** and professor in extending his support to carry out my study.

We are greatly indebted to **Dr. R. Manimegalai, Head of the Department**, Computer Science and Engineering for her guidance which was instrumental in the completion of my project.

We are very grateful to **Ms. P. Priya Ponnusamy**, our **Project guide** for her constant encouragement and support throughout my course.

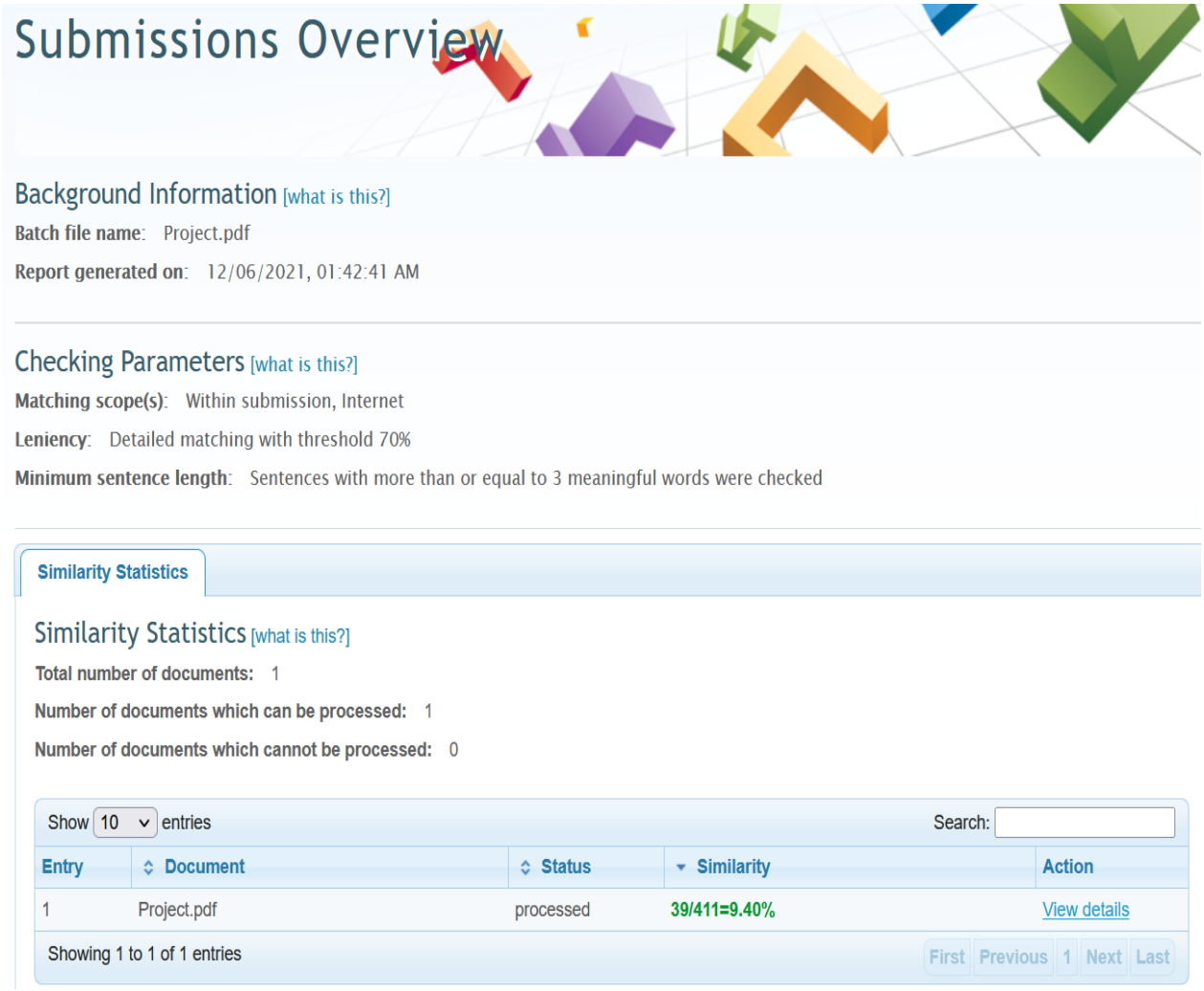
We express our sincere thanks to our project coordinator, **Mr. S. Thivaharan**, for his constant encouragement and support throughout my course.

Finally, we take this opportunity to extend our deepest appreciation to our **family and friends**, who supported us to complete our project during the crucial times.

R. BHUVANESHWARI

M.VARSHINI

PLAGARISM REPORT



Submissions Overview

Background Information [what is this?]

Batch file name: Project.pdf

Report generated on: 12/06/2021, 01:42:41 AM

Checking Parameters [what is this?]

Matching scope(s): Within submission, Internet

Leniency: Detailed matching with threshold 70%

Minimum sentence length: Sentences with more than or equal to 3 meaningful words were checked

Similarity Statistics

Similarity Statistics [what is this?]

Total number of documents: 1

Number of documents which can be processed: 1

Number of documents which cannot be processed: 0

Show 10 entries Search:

Entry	Document	Status	Similarity	Action
1	Project.pdf	processed	39/411=9.40%	View details

Showing 1 to 1 of 1 entries First Previous 1 Next Last

ABSTRACT

The subsequent era of computing is Cloud computing. Nowadays, it is helpful in public as well as private organizations. Cloud computing is a set of servers that service the needs of clients from multiple locations.

Cloud can store a huge amount of client requests with the help of datacentres. A wide range of tools, virtualization options, and an integrated production platform are offered by cloud infrastructure. Cloud computing allows users to use a large range of computational tools.

With the number of internet users, it is impossible to fulfill all the user demands. This issue can be solved with optimal load balancing with the cloud service provider. Load balancing plays a major component in Cloud Computing.

Load balancing prevents overcrowding. It is a technique of distributing workload through several VMs. The VMs are present in the server over the network. It is used to utilize the resource effectively. It also reduces data transfer time and average response time.

The suggested algorithm aims to uniformly spread the load over all servers in a cloud network. The suggested algorithm is then optimized and the results are less than the suggested algorithm. Response time and data processing time have been reduced using the proposed methods which is better than the existing systems.

LIST OF TABLES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
DC	Datacentre
ESCE	Equally Spread Current Execution
IaaS	Infrastructure As A Service
ICT	Information and Communication Technology
IDE	Integrated Development Environment
PaaS	Platform As A Service
R	Region
RR	Round Robin
RTT	Round-Trip Time
SaaS	Software As A Service
VM	Virtual Machine
VMM	Virtual Machine Manager

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 HISTORY OF CLOUD COMPUTING

Cloud computing has improved its existing technologies in the context of the services they provide. Grid computing which is a distributed computing was not flexible when compared to cloud computing. Various attractive features have resulted in the increment of cloud computing. The *Pay-as-you-go* model is followed by Cloud Computing. On-demand provisioning of the resources is provided by Cloud Computing.

The rising standards have increased the demand for cloud computing within the field of business where infrastructure is high. Using cloud technology, one can have runtime environments, infrastructure, and services. Users from the different domains can make better advantage of cloud computing as per their needs. The main entity is that the cloud provider which enables the users to use different services as per their needs. They provide purchasers with many resources like network, storage capacity which may be joined with any registration process where any user who is authenticated can work and request services of the cloud.

Among the varied different tasks of providers of cloud services, the crucial one is to assign nodes for requests of users. The above task needs to be more efficient. To perform this, the total time interval should be less [26] as possible whereas in parallel various other problems with network delays and heterogeneity got to be managed. Cloud computing is popular within the field of Information and Communication Technology (ICT).

Rapid increment in the cloud requests, providers got to build more capabilities in the components which may be performed by increasing their count or power or both. This example may become an outsized network consisting of cloud users, nodes, tasks and Virtual Machines (VMs). As the network size becomes large, workload also got to be managed to increase overall performance. This area of load balancing aims at the increased detection rate and the distribution of work in the network.

1.2 OBJECTIVE

The major objective is to generate an effective mechanism for load balancing, which may be a benefit for both the service providers also as a cloud user. Proper distribution of the load among prevailing nodes in the network are going to be the simplest thanks to utilize the resources and to process the requests of user. This task of load balancing is often performed by scheduling of tasks into VM then from VM to nodes existing in system. An effective algorithm for Load balancing will get high rate of fault tolerance.

1.3 CHALLENGES AND MOTIVATION

Cloud computing is earning more attention from the users as it is the scalable, elastic, easily accessible, and least delivery of services. These features contribute in making it marketable. This model may set a replacement world to cloud computing where every user can run its services as per requirements. The working of the interior process is well known by cloud computing.

The overall work of the complete procedure of cloud computing is directly proportional to the number of users accessing the services. It is a challenging task to manage the resources like memory, CPU and secondary storage. This three are

the prerequisite of the cloud service to provide load balancing in an effective manner. Traffic will also rise exponentially.

1.4 PROBLEM STATEMENT

Cloud is an active environment. The load balancing algorithm plays an important role in distributing the request of the client to the VMs. Existing algorithms are very complex in nature, static, cost inefficient, less throughput and it takes more time for responding to the VM and processing the request. There is a need to find a new load balancing algorithm which is more efficient in cost and also in time.

1.5 PROJECT OVERVIEW

There are many current load balancing algorithms. Some of them are

- Round Robin Load Balancing Algorithm
- Equally Spaced Current Execution (ESCE) Load Balancing Algorithm
- Throttled Load Balancing Algorithm

The proposed algorithm extends the benefits of ESCE and also dynamically allocate the requests to the VMs and also it is time efficient. After implementing the algorithm, the datacentre processing time and response time is compared with the existing algorithms. The comparison of the algorithms with different userbases and different datacentres were plotted on the graph.

CHAPTER 2

CLOUD COMPUTING AND LOAD BALANCING

2.1 CLOUD COMPUTING

Cloud Computing has been described in variety of methods with the aid of several researchers. The meanings of cloud computing are given based on their own interpretations. The below mentioned two are most known definitions:

- **NIST**

  Cloud Computing is model for enabling convenient, on-demand network access to the shared pool of the configurable computing resources. For Example, Servers, Networks, Services and Storage which should be provisioned and released with the minimal management effort or a service provider interaction [40].

- **Foster**

  A large-scale distributed computing paradigm that's driven by economic scale, during which pool of abstracted virtualized, storage, dynamically-scalable, platforms, managed computing power and services are delivered to external customers on demand over Internet [41].

2.2 BENEFITS OF CLOUD

The expenses at the initial and recurring are lower than traditional computing. Cost is reduced. Massive infrastructure is offered by cloud providers to us. A large amount of data can be stored and maintained. Sudden workloads are also managed effectively. It can also be flexible. It additionally continues machine stability.

2.3 COMPONENTS OF CLOUD COMPUTING

Components of cloud work as the integrated unit which comprises of datacentres, servers and clients. The Figure 2.1 describes the cloud components.

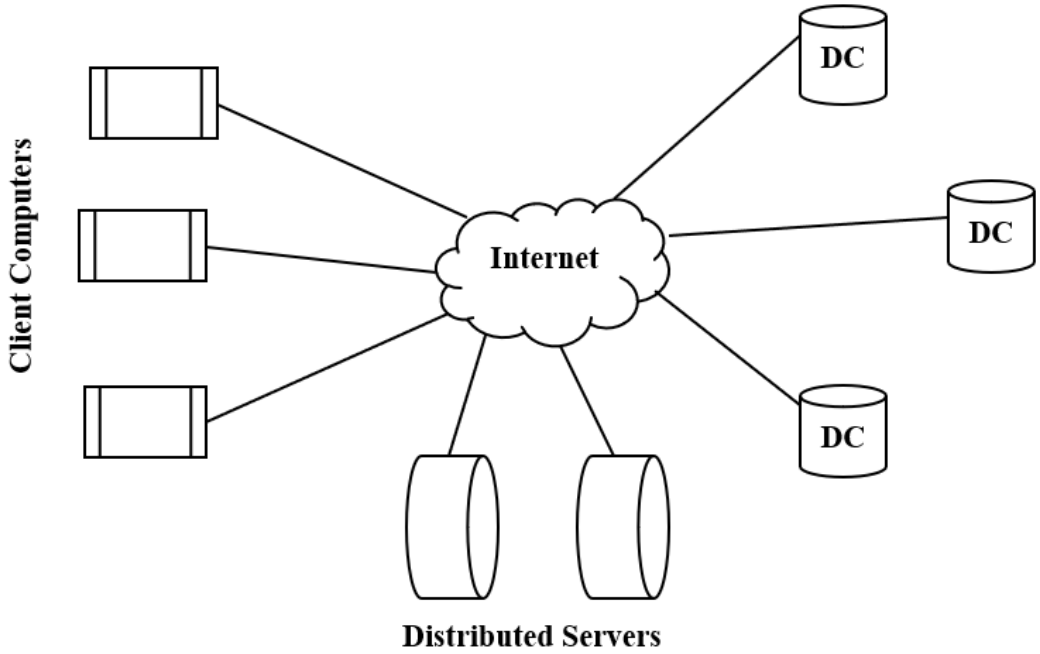


Figure 2.1 Components of cloud computing

Each component works specific role demanded by user. They communicate in overall network according to requirements.

- **Clients**

  Computers use in our life are included in clients. It consists of desktop computer, mobile phone, tablet. The clients gain access to use cloud services through internet.

- **Data Centres**

  The most core element of entire cloud process. It consists of many nodes in the room. Service providers takes care of Configuration.

- **Distributed Servers**

  These servers are called as nodes. It need not fit in same location as user. Distributed Servers helps in increment of the fault tolerance rate.

2.4 CLOUD COMPUTING ARCHITECTURE

Cloud computing supports IT service which will be consumed as a utility and delivered through a network. It is possible to organize each concrete realizations of cloud computing into a layered view. It covers the entire stack from hardware appliances to software systems.

The whole architecture can be understood by studying separately each layer in architecture. The layers are restricted to work for specific tasks. The Figure 2.2 depicts the layered architecture of cloud computing with several layers.

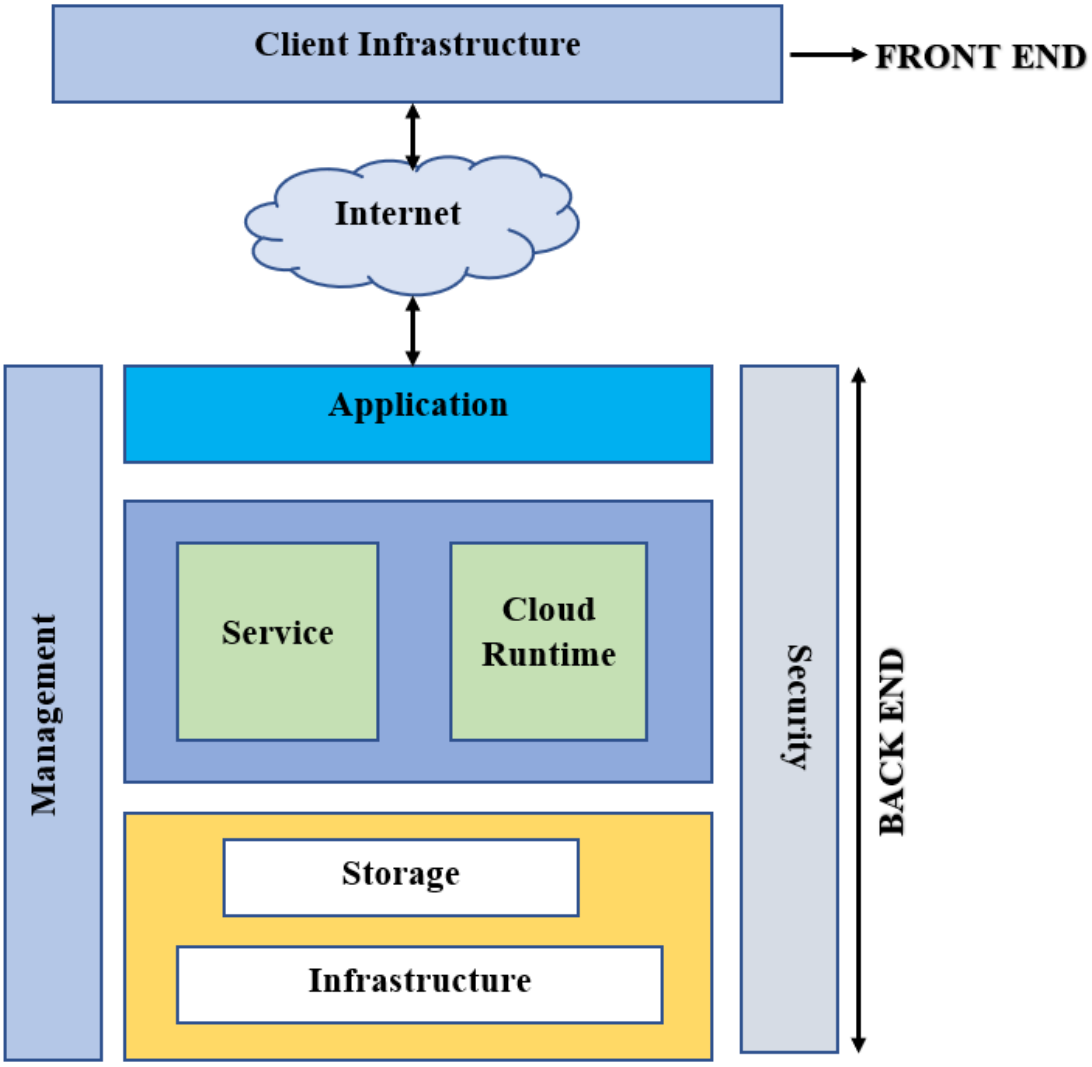


Figure 2.2 Cloud computing architecture

2.5 CLOUD SERVICE MODELS

Once cloud is developed and ready to be used by the customers, it should be deployed. As shown in the Figure 2.3, a service can be delivered to the cloud users through Software, Platform and Infrastructure.

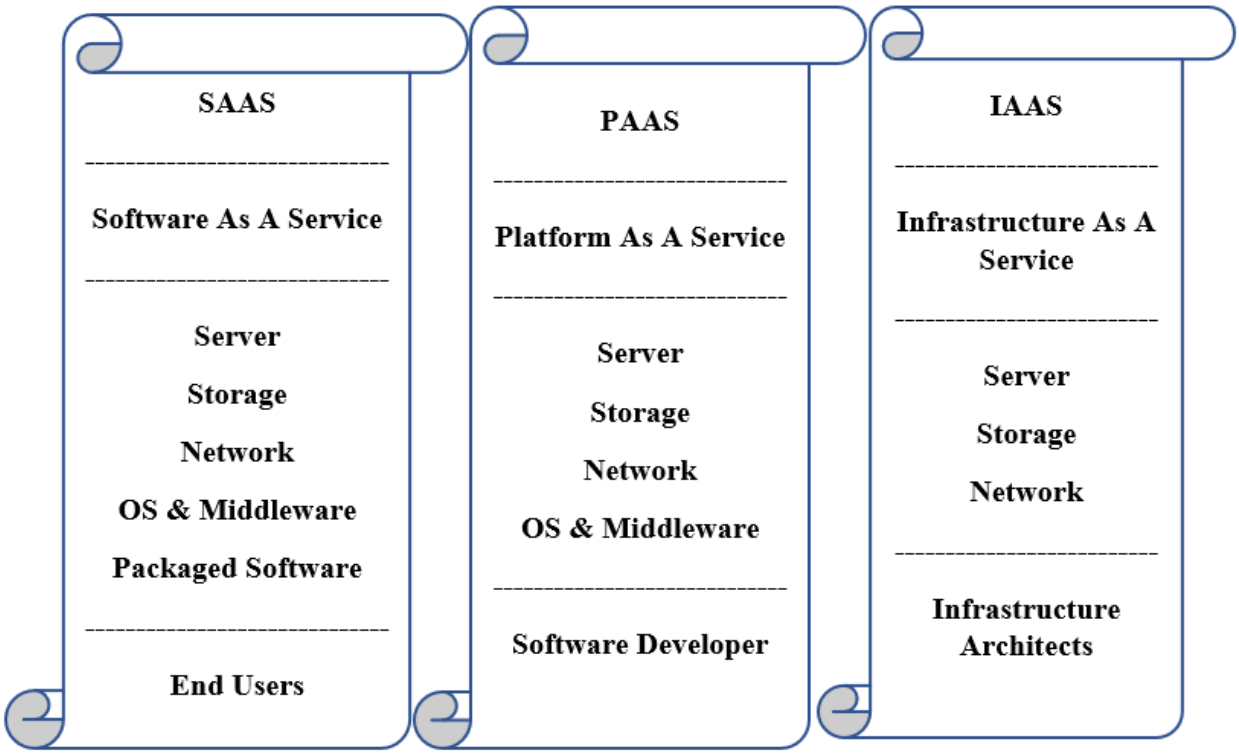


Figure 2.3 Cloud service models

The cloud provider models are of three types. They are

- **Software as A Service (SaaS)**

  SaaS is software delivery model. It facilitates the user with variety of applications like social networking. The Figure 2.4 depicts the working of SaaS. Facebook, NetSuite and Google Docs are examples of SaaS.

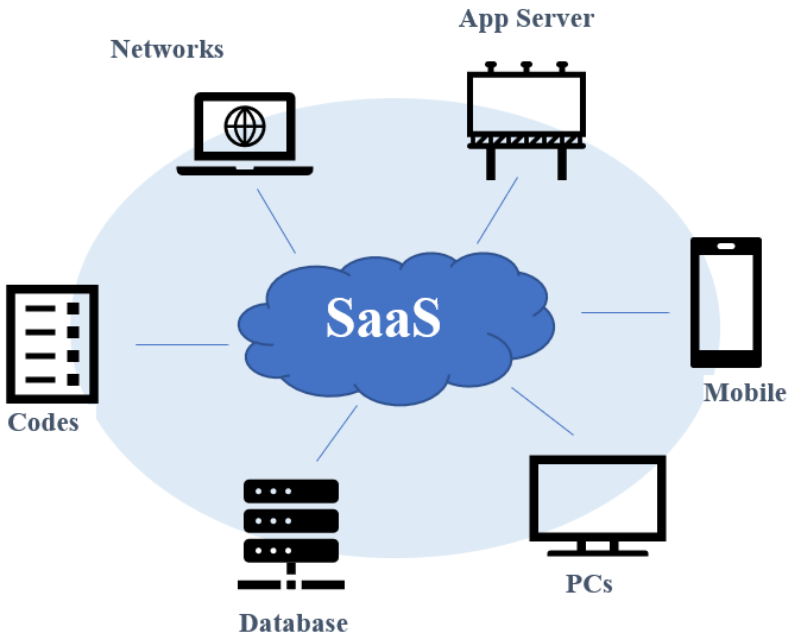


Figure 2.4 Software As A Service

❖ **Platform as A Service (PaaS)**

PaaS provides facility to the users for developing their own applications. It also provides a generalized development environment like on demand security and scalability. The Figure 2.5 depicts the PaaS. Examples of PaaS are Google App Engine and Windows Azure.

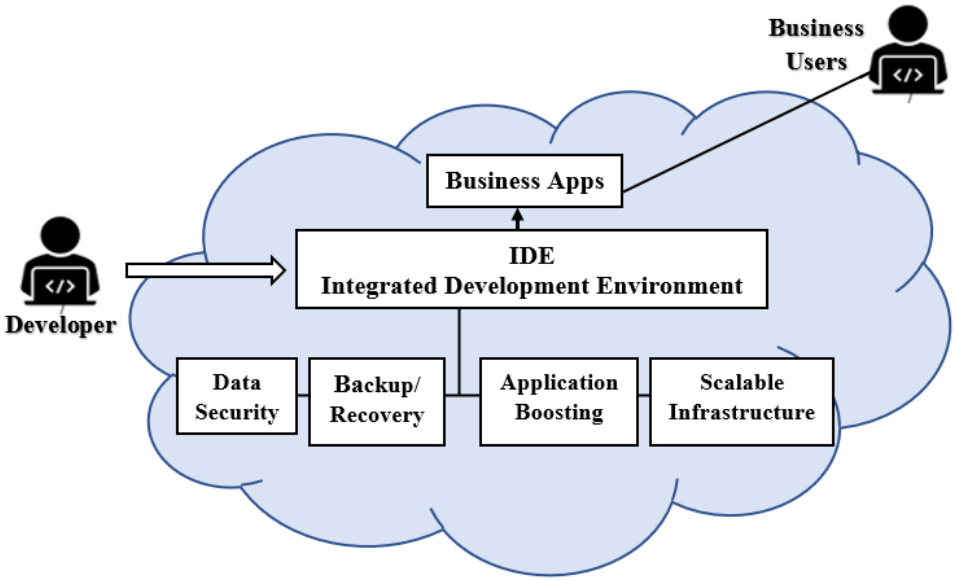


Figure 2.5 Platform As A Service

❖ **Infrastructure as A Service (IaaS)**

IaaS or Hardware as a Service (HaaS) offers facilities to users like customizing infrastructure as per the requirements. It can process huge amount of data over a huge network. Working of the IaaS in the cloud environment is shown in the Figure 2.6. Users can deploy and run any software. Examples of IaaS are Amazon and Go Grid.

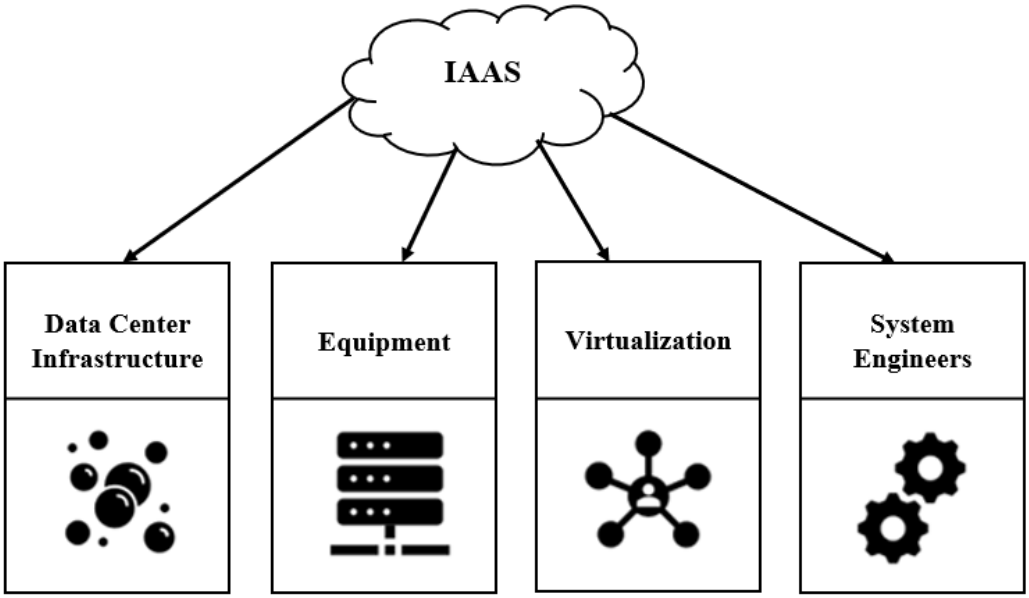


Figure 2.6 Infrastructure As A Service

2.6 CLOUD DEPLOYMENT MODELS

Deployment models outline the kind of right to entry to the cloud. There are four deployment models. They are

- ❖ Community Cloud
- ❖ Public Cloud
- ❖ Private Cloud and
- ❖ Hybrid Cloud

There are a number of cloud consumers who want the infrastructure of different sizes. It identifies boundary within which cloud computing services are implemented. It also provides hint on underlying infrastructure adopted to support such qualifies and services them. Cloud deployment models are depicted in the Figure 2.7.

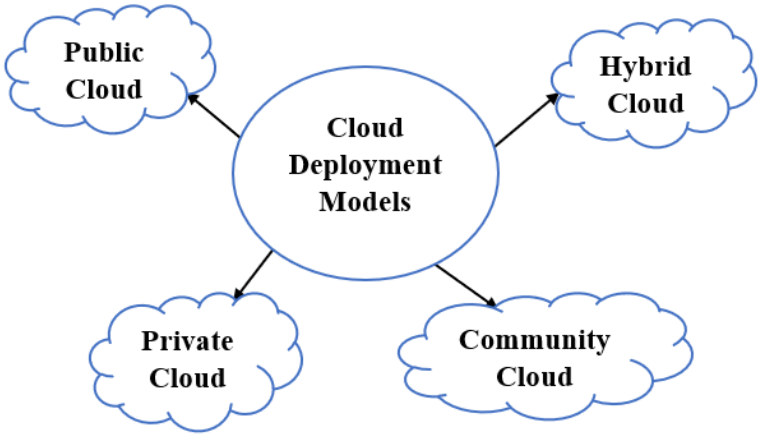


Figure 2.7 Cloud deployment models

2.7 VIRTUALIZATION

Virtualization is a technique that permits sharing single physical instance of an application or resource among multiple organizations or tenants that is customers. The basic virtualization of the VM in the cloud computing has been displayed in the Figure 2.8.

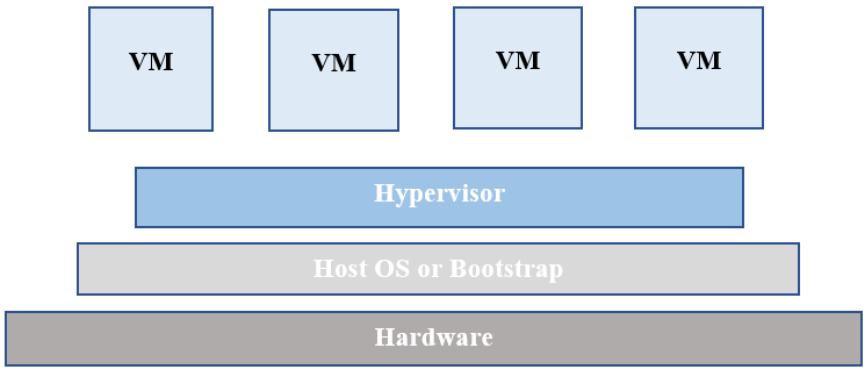


Figure 2.8 Virtualization

There are three styles of virtualization. They are

- **Full Virtualization**

  Primary hardware is absolutely simulated in the complete virtualization. Guest software program does not require any modification in software program to run.

- **Emulation Virtualization**

  VM simulates the hardware. Guest OS does not require modification.

- **Para- Virtualization**

  Hardware isn't always simulated through the VM. Guest Software runs their very own isolated domains.

2.8 LOAD BALANCING

Load balancing [39] is an essential characteristic in cloud computing. It is a way of distributing workloads to multiple resources across one or more servers. It reduces the cost and increases the flexibility of the load. It ensures maximum throughput in minimum response time. The basic way in which the load balancing in cloud computing occurs is shown in the Figure 2.9.

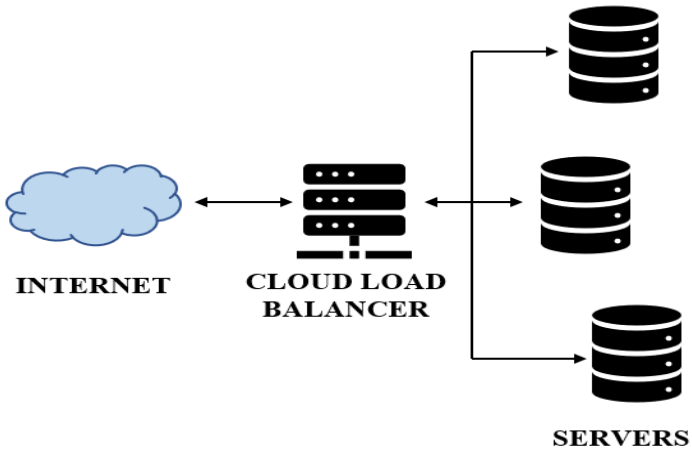


Figure 2.9 Load balancing in cloud computing

2.9 GOALS OF LOAD BALANCING

Various organizations like IT and Engineering have own networking systems and use customized policies for load balancing in their network. They use global policy for cloud environment and specify technical goals for load balancers which compliment business goals. It helps to improves the performance It improve resource utilization. Fault is tolerated by this system.

2.10 LOAD BALANCING EXECUTION

A cloud network consists of various nodes which consist of processing units and storage units. Every node has many VMs to serve requests and a task is assigned to any of VM or resource.

Resource is released after completing a task and can be allocated again to any other request according to some allocation policy. CloudSim [37] can effectively simulate load balancing or scheduling in cloud environment. Figure 2.10 shows the hierarchy and assignment of cloud computing.

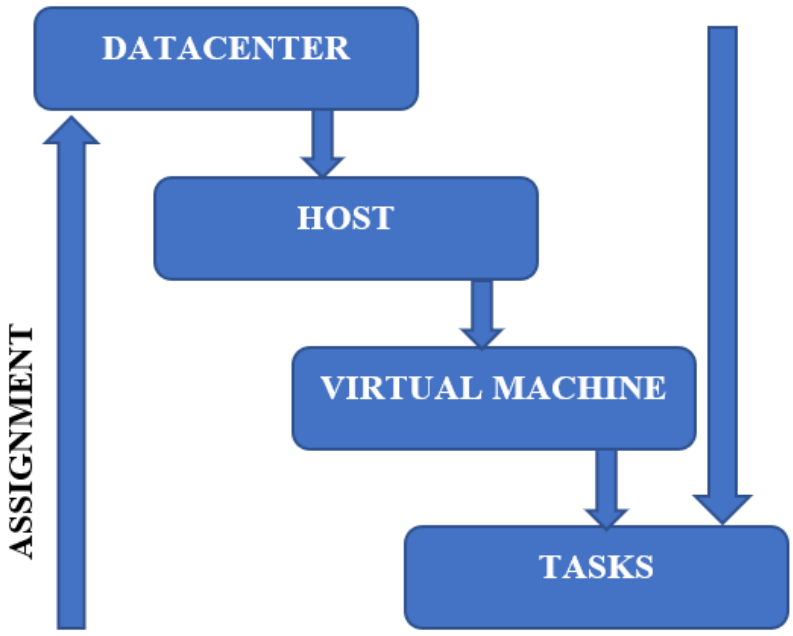


Figure 2.10 Hierarchy and assignment of cloud components

Task scheduling and resource provisioning are main mechanism which is responsible for proper load balancing in cloud. Resource provisioning defines resource mapping with tasks or with any other network entity. The Figure 2.11 demonstrated the execution of load balancing algorithm in the cloud environment.

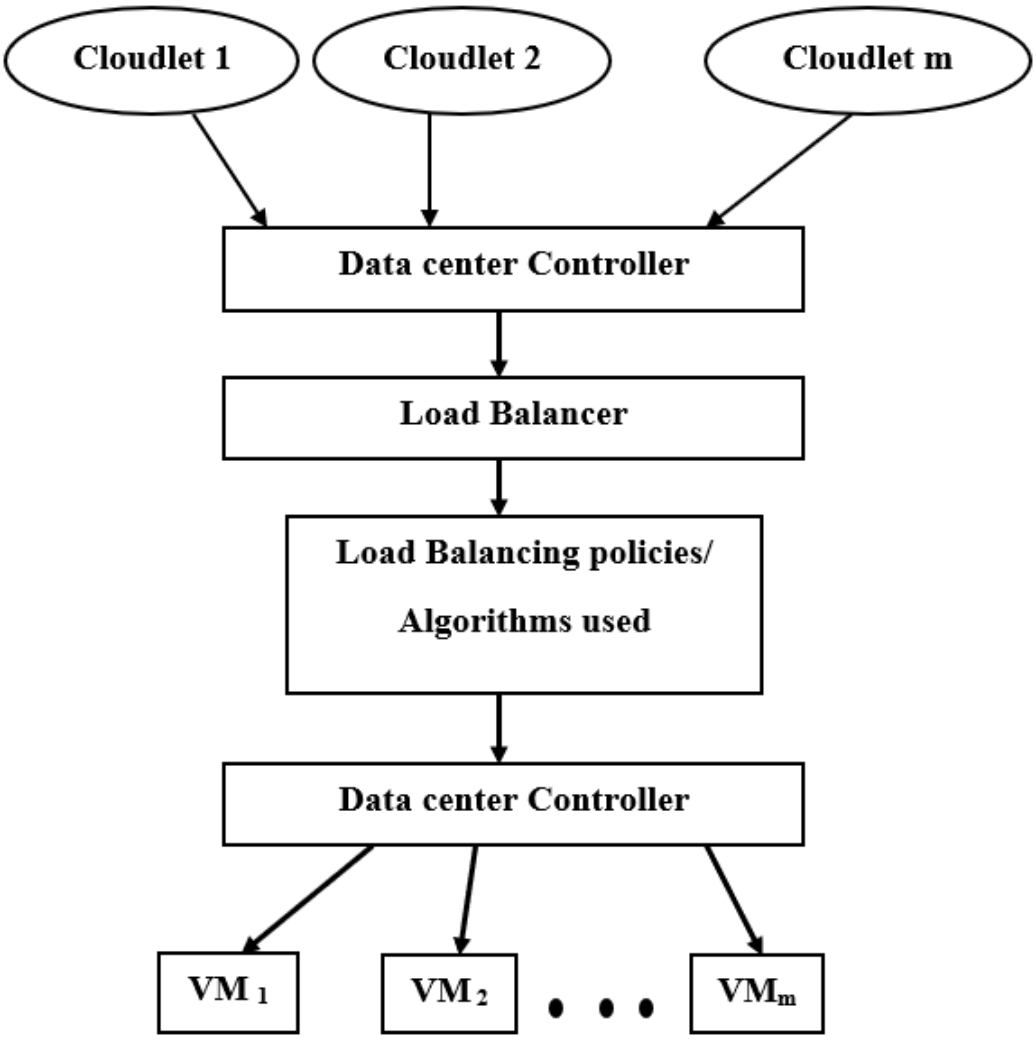


Figure 2.11 Execution of load balancing algorithm

2.11 PERFORMANCE METRICS FOR LBA EVALUATION

Load balancing algorithms are measured by many metrics to evaluate the performance and the working. Those metrics are defined in short in this section. The Figure 2.12 depicts the load balancing parameters in cloud computing.

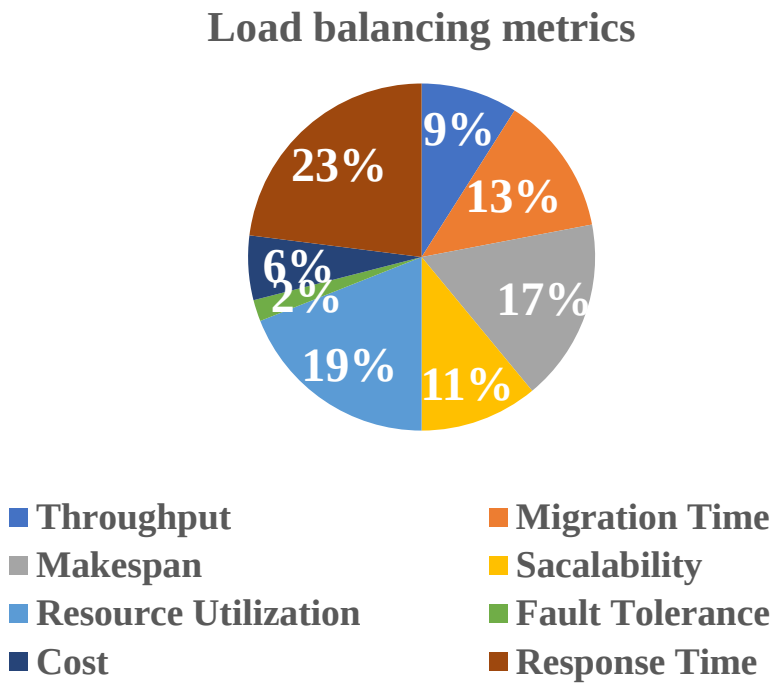


Figure 2.12 Load balancing metrics in cloud computing

❖ **Throughput**

This metric is used to calculate the entire range of tasks, whose finishing touch has been completed successfully. For normal gadget performance, excessive throughput is needed.

❖ **Overhead**

Overhead mixed with any load balancing set of rules prefaces the greater price engaged in imposing the set of rules. Overhead associated specifies the quantity of overhead involved while completing a load-balancing algorithm. It covers overhead because of motion of tasks; inter-method contact and inter processors. It needs to be as little as possible.

❖ **Fault tolerance**

Measuring the potential of an algorithm to perform equal load balancing by chances of any shortcoming. It ought to be pretty faulted.

❖ **Migration time**

It is defined as, the complete time wanted in migrating the assets or jobs from one node to another. It must be decreased or minimized.

❖ **Response time**

Response time may be decided as, the meantime among sending a request and getting its response. To increase the general performance, it must be minimized.

❖ **Datacentre processing time**

It is the time taken for the datacentre for processing the request. To get higher performance, datacentre processing time should be reduced.

❖ **Resource utilization**

It is used to guarantee the correct usage of all the ones resources, which included the complete system. This difficulty should be optimized to have a green load balancing algorithm.

❖ **Scalability**

It is the ability to carry out uniform load balancing in a device with the upward push within the wide variety of nodes, in step with the requirements. Higher scalability is preferred.

❖ **Performance**

It is used to analyze how green the gadget is. This must be uplifted at an affordable cost. For example, decreasing the reaction time though maintaining the receivable delays.

CHAPTER 3

LITERATURE SURVEY

3.1 EXISTING SYSTEM

Cloud Computing is the current advancement in the technology field which is related to IT industry. In [34] M. A. Bhagyabini et al provided review of the load balancing techniques. On the basis of system topology and load, the algorithms are classified. Multiple examples and their implementation and Key technologies was specified in the paper. Challenges are also mentioned by the authors in this paper. Nidhi et al [22] mentioned strategies that objectives to lessen the overhead, reaction time and overall performance in the paper. It also analyzed parameters governing performance.

Shu-Ching et al [31] combined the functionalities of Load balancing Minimum-Maximum and Opportunistic Load balancing in order to propose scheduling algorithm and then it gave improvement to existing Min-Max algorithm. OLB saved each node busy at the same time as now no longer regarding the workload within the unique node. OLB assigned responsibilities in random way to all the available nodes while MCT algorithm assigned responsibilities to most effective that node which predicted minimal finishing touch time to different nodes.

Another very important and effective algorithm came into existence by Che-Lun Hung et al in [7] which was LB3M. Gave the concept to get any resource of cloud by incorporating Co-operative Scheduling of power. It was a good replacement to challenges concerned to load balancing keeping into concern energy efficiency. It incorporated both centralized as well as distributed approach making the best use of inherent efficiently related to centralized one and energy efficiency and fault tolerant behavior of distributed method. [13] Percentage of

node utilization in PALB is calculated and this percent decided the count of computed nodes that are kept operated when another one gets shut down completely. The technique consists of sub areas: Balancing area gave method to instantiate the VMs based on utilization percentage. Another area which is Upscale gave method to power on the nodes. Area of Downscale was used to shutdown the nodes that are idle. It gave promise to decrease consumption of power and managing the resource availability in parallel.

Raul Calvo et al [27] proposed method to manage image collection of large amounts in real world techniques. This paper creates service and provide its use for analysis of images. Various operations on data are stated to work in distributed area for different sub- images. Now these sub-images will be stored and their processing is done separately by multiple agents. It deals with the execution of method in large images in parallel way. Resource scaling and consuming power are the factors to be dealt with any load balancing method. [35] Improvements that can be measured are obtained in the cloud environment. Load balancing techniques should be such that to obtain measurable improvements in the resource utilization and availability of cloud computing environment.

Alexandro Iosup et al [2] studied the services of cloud and analyzed for scientific computing. They experimented on workloads of real scientific world of MTC requests of user. MTC stans for many task computing. They deal with applications that are loosely coupled which complete tasks as per scientific goals. Srinivas [31] proposed the algorithm for load balancing algorithm for load balancing adding fuzzy logic to its computation. It made the use of speed of processor and incorporated assigned load onto overall load in cloud environment [17]. Here authors proposed a new logic for dynamic balancing of load in cloud environment with parameters like disk space and status of VM and then stated Fuzzy Active Monitoring Load Balancer (FAMLB).

Milan E Soklic [18] et al focused his work in the elaboration of the load balancing techniques. They include static, round robin method, shortest queue in cloud environment. The result of the experimental analysis is that it states that in the dynamic environment diffusive load balancing is better and efficient as compared to static load balancing [3]. Gave a difference between strategies that exists among various methods and algorithms.

Parallel processing [14] can be seen in a network processor. It comprises of many on-chip processors which is used to perform operations of packet level. It assures high throughput by providing fine load balancing to the available processors. It may also have a disadvantage of high rate of out of order packets. Scheduling packets is done by ORR which is Ordered Round Robin. The heterogeneous processor which gives loads need to be handled accurately. The processor will load which are from the processors are perfectly ordered. Analysis of the derived expressions and the throughput in the terms of batch size, count odd processors scheduled and scheduling time.

Jaspreet Kaur elaborated active virtual machine algorithm so as to urge free and appropriate virtual machine in lesser time. She performed simulation to try to a comparative study of round robin and to spread execution policies concerned to load balancing for lesser time and price. Algorithm in [36] added capacity associated with dynamic balancing method of cloud. Experiments performed by Zhang Bo was clearly demonstrated that achieved better degree for load balancing and took lesser time in loading tasks.

Soumya Ray et al [30] researched many algorithms like central queuing algorithm. The analysis was carried between MIPS vs HOST and MIPS vs VM. The experiment demonstrated that the response time can be improved in the terms of number of VMs in Datacentre. In order to handle the random selection-based load distribution problem, dynamic load balancing algorithm are often implemented because the future course of labor to gauge various parameters.

The author [15] proposed algorithm for load distribution of workloads among nodes of cloud, by utilization of Ant Colony Optimization. This algorithm uses the concept of Ant Colony Optimization. Shridhar G. Domanal and G. Ram Mohana Reddy et al [28] have proposed a local optimized load balancing approach for distributing incoming job request uniformly among the servers or virtual machines. Performance of proposed algorithm is analyzed using Cloud Analyst [5] simulator, further comparison is done with another existing Round Robin and Throttled Algorithm.

In [32], the authors have analyzed various policies utilized with different algorithm for load balancing using a tool called Cloud Analyst. Dynamic Round Robin algorithm [1] is the improvement over static Round Robin algorithm, this paper is well analyzed and Dynamic Round Robin algorithm with varying parameters. Ching- Chi Lin et.al [7] have proposed new Dynamic Round Robin Algorithm energy – aware VM scheduling and consolidations.

3.2 DRAWBACKS IN EXISTING SYSTEMS

Though there are many advantages in the existing algorithms, there are some disadvantages.

- The state of previous allocation of VM is not saved.
- Existing loads on resources is not considered
- Pre-emption is required.
- Some algorithms are static.
- Starvation is more. Smaller tasks waits until the larger one completes.
- Network overhead occurs.
- Complex in nature.

CHAPTER 4

PROPOSED METHODOLOGY

4.1 LOAD BALANCING ALGORITHMS

Various load balancing algorithms have been evolved and are in use since region of distributed computing. Load balancing is based on many factors like system configure and system topology and old techniques can be used in cloud computing for improved resource utilization. The Figure 4.1 depicts all types of load balancing algorithms.

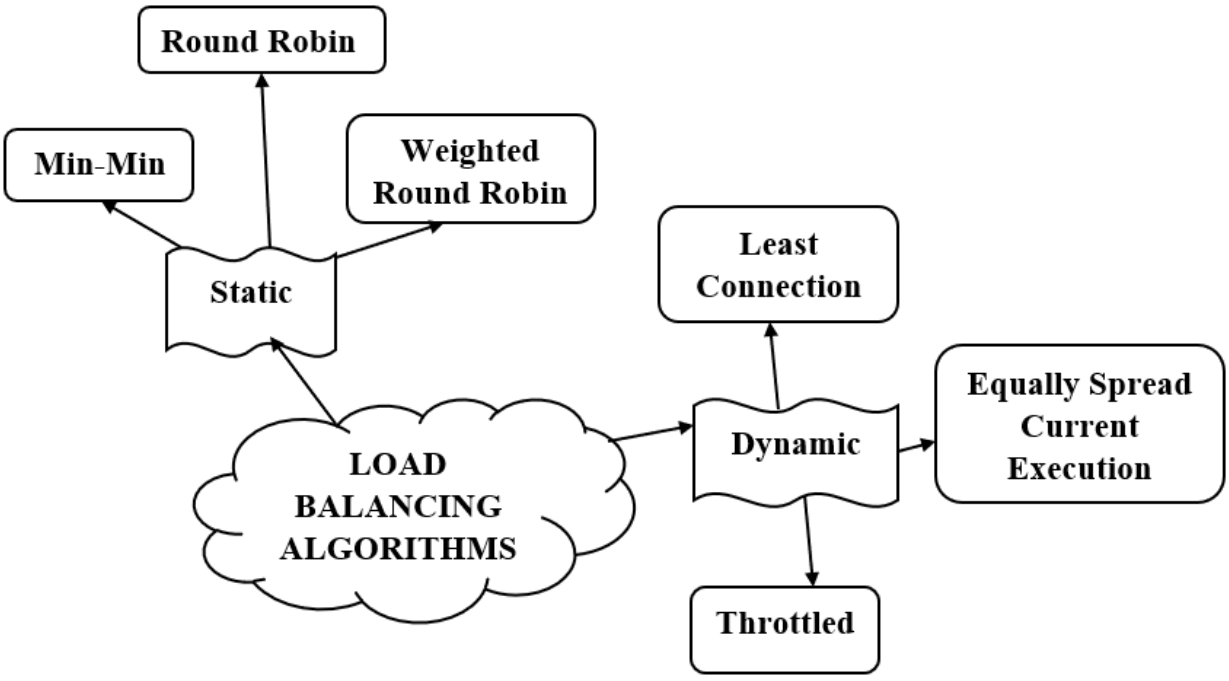


Figure 4.1 Types of load balancing algorithms

Basically, load balancing techniques are classified into two categories

- ❖ Static load balancing
- ❖ Dynamic load balancing

4.1.1 STATIC LOAD BALANCING TECHNIQUES

Static techniques are easy to define. It is feasible for little and glued request handling network. Static or deterministic techniques make a system less scalable and can't handle changes in nature of tasks. It requires more manual effort for mapping more users to system resources. The working of Static Load Balancer is explained in the Figure 4.2.

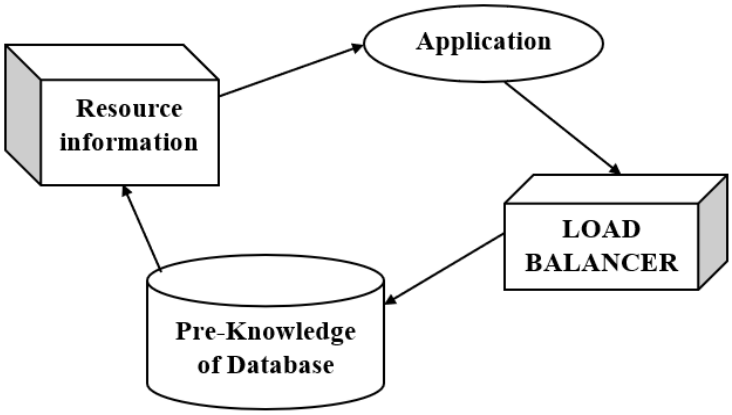


Figure 4.2 Static load balancer

- **Round Robin**

  It handles clients request during a circular manner and each task get lock on VM till it get completed and serves user tasks on FIFO basis. The algorithm, where the round robin load balancing algorithm works is revealed in the Figure 4.3.

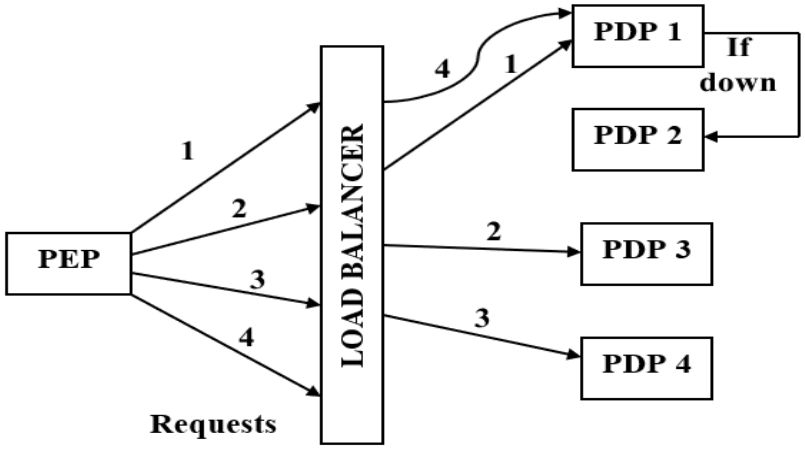


Figure 4.3 Round robin algorithm

❖ **CLBDM (Central Load Balancing Decision Model)**

It is an improvement over round robin and also finds out time duration of session established between client and node. Time Duration can be defined as execution time taken by task till completion on given VM or resource. The Figure 4.4 depicts CLBDM load balancing.

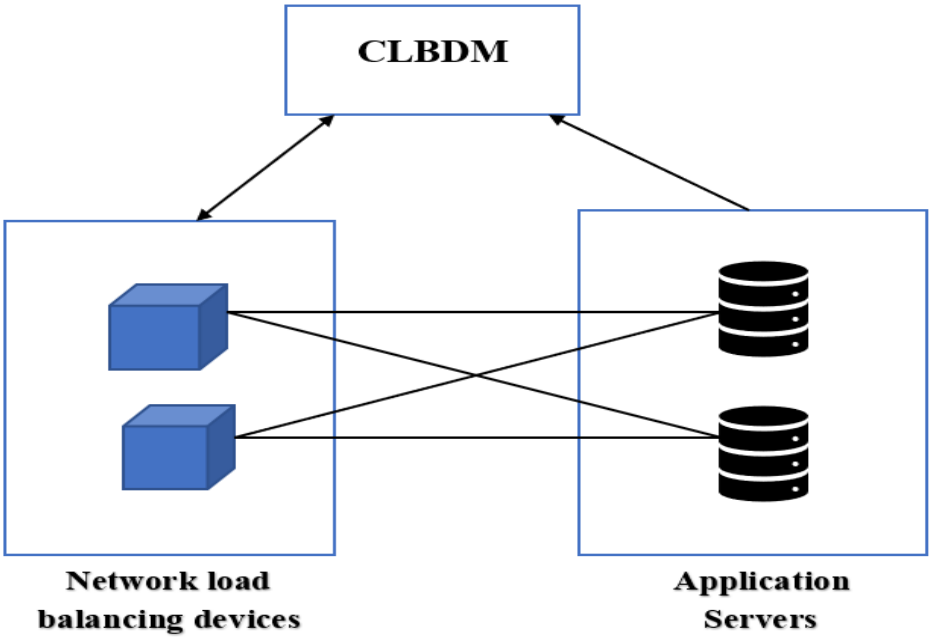


Figure 4.4 CLBDM load balancing algorithm

4.1.2 DYNAMIC LOAD BALANCING TECHNIQUES

Dynamic techniques can also be classified in centralized or distributed topology. All decisions for system resource provisioning or scheduling are controlled by a master and to a slave means slaves merely serves request forwarded by master to them and cannot take their own decisions.

Whereas in distributed system, control is distributed in network and more than one node can handle provisioning or scheduling of resources. The Figure 4.5 displays the process of Dynamic Load Balancer.

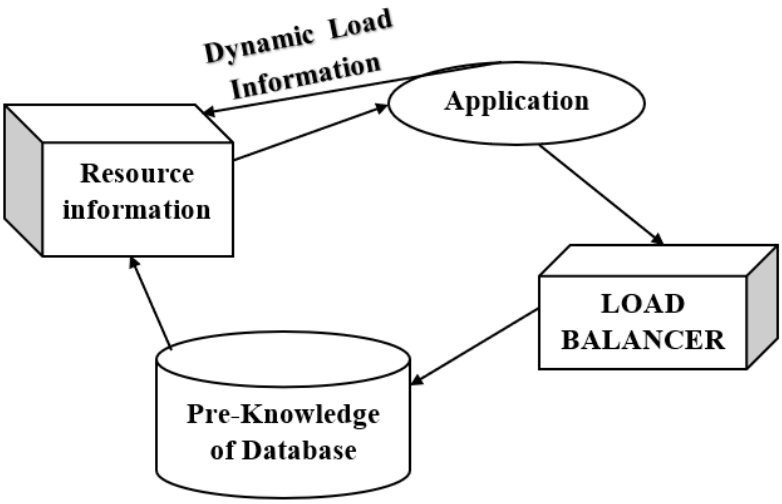


Figure 4.5 Dynamic load balancer

Commonly used dynamic policies are:

- **ESCE Policy**

  It equally spreads execution load to all or any available resources and maintain record of all resources and number of tasks assigned to each resource or VM when any new task is received to load balancer it searches for most underutilized resource or VM.

- **Throttled Policy**

  Under this policy, each resource is assigned with only one task at a time and any other job can be assigned only after completion of previous task assigned to it. The useful resource lists are maintained through the Load Balancer.

  When a new user task is received job, manager check list and if any free resource is not found, it keeps task in waiting queue and poll on list to check VM status. The Figure 4.6 describes about the flowchart of Throttled load balancing algorithm.

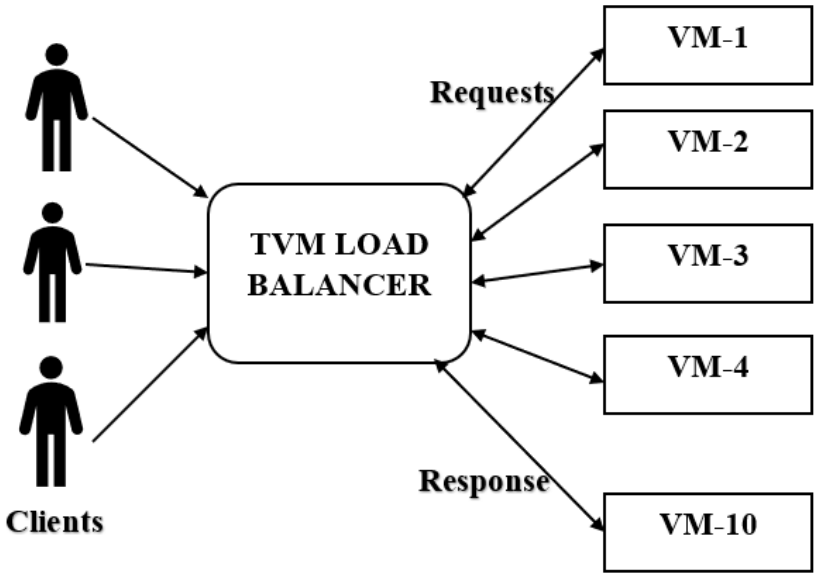


Figure 4.6 Throttled load balancing algorithm

4.2 STEPS FOR IMPLEMENTATION

The steps for implementing the Hash ESCE and Optimized algorithm are as follows.

- **Creation of Virtual Cloud using JAVA**

  Virtual Cloud is created by JAVA IDE (Eclipse). GUI is the JAVA interface. The configuration of cloud runs at the Eclipse. Cloud Analyst workspace is used to create the cloud.

- **Configuring the Cloud Environment**

  Cloud Environment is configured using JAVA by configuring Bandwidth, Datacentre Configuration and Latency Matrix.

- **Creation of Datacentres**

  Processing the requests of the users using different parameters, the datacentres are created.

- **Creation of User Bases**

  Sending the request using different parameters to Datacentres creates the userbase.

- **Service Broker Policies Implementation**

  There are three service broker policies. They are

  - Closest Datacentre (CDC)
  - Optimize Response Time (ORT) and
  - Reconfiguring with the dynamic load (RDL).

- **Implementation of load balancing algorithm**

  The Hash ESCE algorithm is implemented.

- Performance analysis based on overall response time and datacentre processing time.

4.3 DESCRIPTION OF HASH ESCE ALGORITHM

The proposed algorithm is an extension of ESCE algorithm. It includes the benefits of ESCE algorithm. The states of VMs are contained in the HashMap. States of the VMs are BUSY and AVAILABLE.

Load balancer will test for the primary available VM when the datacentre gets the request. If VMs have status as available then, the request is allocated. The request within the datacentre is queued, if there may be no available VM.

Then, the Load balancer offers the load to all VMs. Load balancer maintains an index table and a queue. The index table will contain the range of VMs. A queue contains the range of requests assigned to the VM. If no VM is

presently free then the Load balancer checks for VM with minimum load. Then the request is allocated to VM.

4.4 HASH ESCE ALGORITHM

HashMap is used to store the VM list and status of each VM. The Load balancer will take a look at for the status and the available VMs will get allocated. The process of implementing the Hash ESCE is shown in the following steps.

Step 1 : HashMap initially does not contain any entries.

Step 2 : Then all VMs states are mapped to AVAILABLE which are in the VM States list.

Step 3 : A new request from the client is sent to the datacenter controller.

Step 4 : Next allocation of the tasks is done by the Load Balancer which is asked by the Datacenter controller.

Step 5 : The request is allocated to VM, if the HashMap list size < VM state list size. Else, the request waits for the VM to get free.

Step 6 : After VM processing the request, the DC controller notifies the LB to release the VMs.

Step 7 : Load balancer updates the state of the VM in both the lists.

Step 8 : When all VM gets busy and the Hash ESCE check for the VM with minimum load and the request is assigned to that machine.

The proposed algorithm is coded in JAVA and the Integrated Development Environment (IDE) used is Eclipse. The cloud simulator used is Cloud Analyst.

The Cloud Analyst is imported in the Eclipse and the algorithm is implemented as per the flowchart as shown in the Figure 4.7. Then, the simulation of the code is run by the IDE. Thus, the simulation page is created.

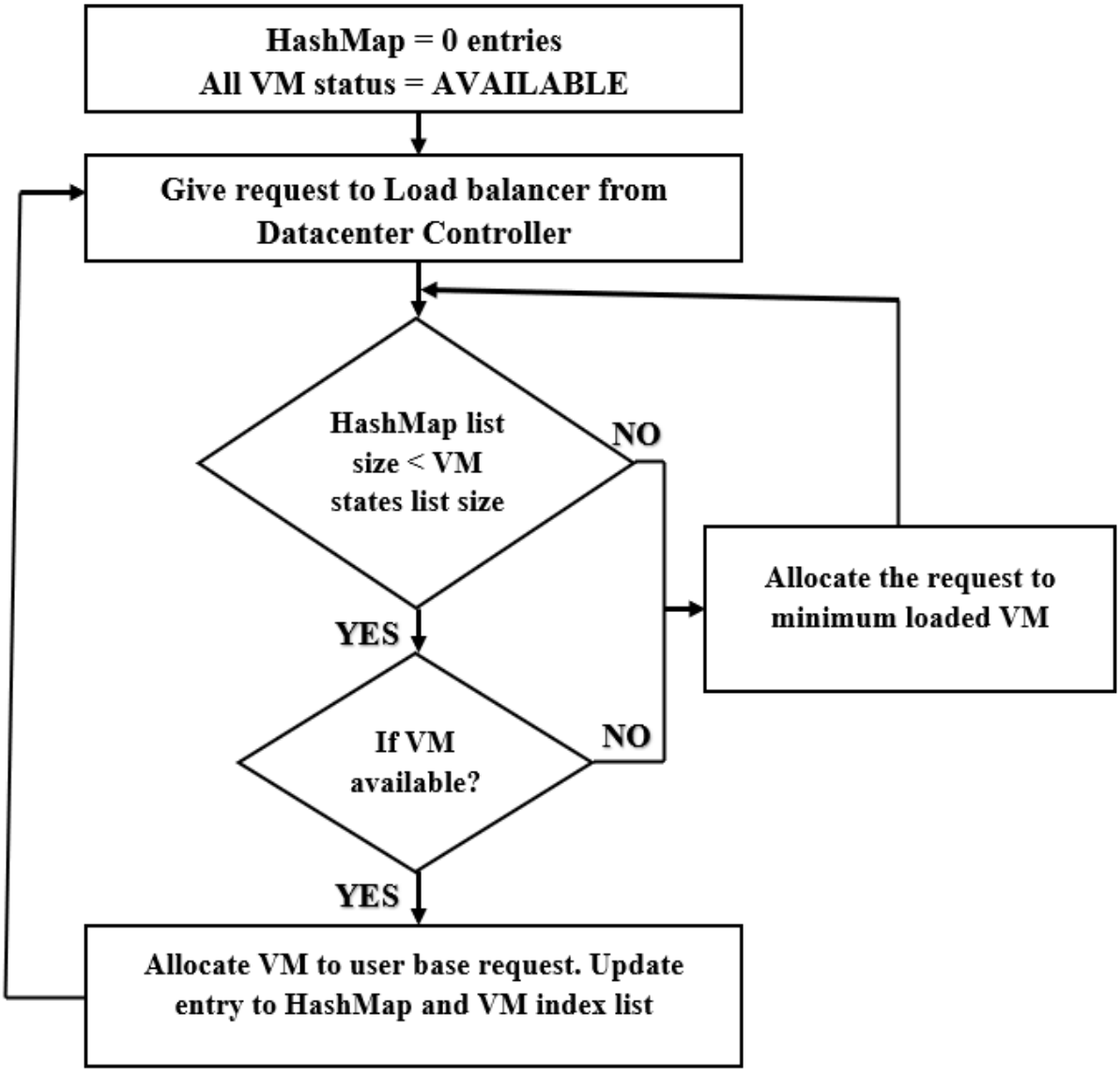


Figure 4.7 Flowchart of hash ESCE algorithm

4.5 DESCRIPTION OF OPTIMIZED ALGORITHM

The optimized algorithm consists of the HashMap which contains the VMs and its states and two variables low and high which are used to indicate the limits. If the count of requests in VM is less than low, then it is added to the lownode queue. If the count of requests in VM is greater than high, then it is added to highnode.

Initially, all the count of requests in VMs are set to lownode. If VM requests count is between the limits then the VM with less requests are allocated. If there are no available VMs within the limit, then the request is allocated to the VMs in the highnodes.

4.6 OPTIMIZED ALGORITHM

The optimized algorithm has been coded in JAVA in the Eclipse IDE. The cloud analyst simulator has been imported in Eclipse and the algorithm implemented in the datacenter in the cloud analyst as per the flowchart shown in the Figure 4.8. The steps of implementing the optimized algorithm are shown in the following steps.

Step 1 : Initialize the VMs with their states as AVAILABLE.

Step 2 : Create two queues lownode and highnode. Initially, all VMs are added to lownode.

Step 3 : Create two variable low and high and initialize them with value which are the limits.

Step 4 : If count of requests in VM is less than high, then the request is allocated automatically.

Step 5 : If count of requests in VM is greater than high then the VM is added to the highnode queue.

Step 6 : If the count of requests is within the limits, then if VM is available the request is allocated. Otherwise, the request is allocated to VMs with minimum requests.

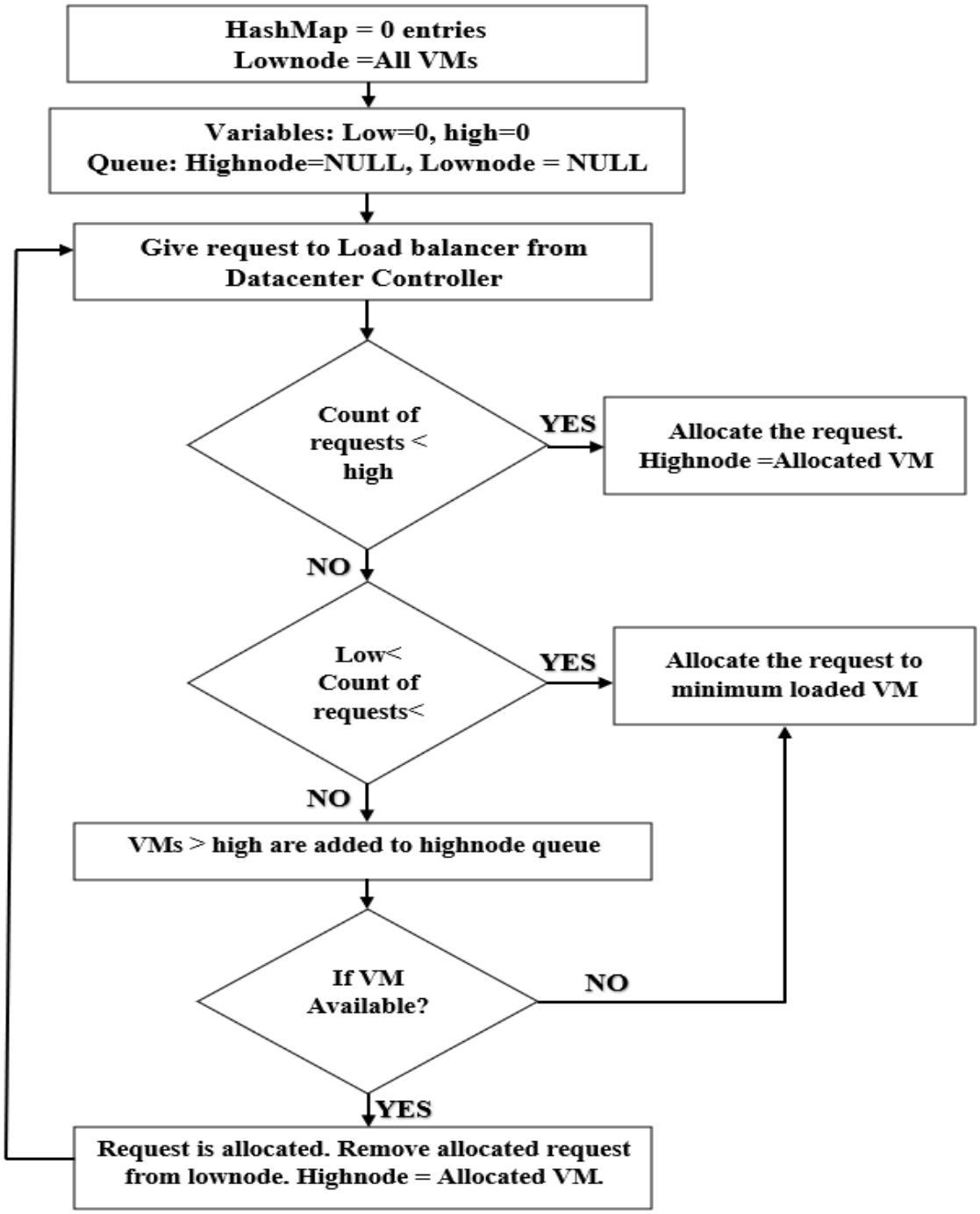


Figure 4.8 Flowchart of optimized algorithm

4.7 ADVANTAGES OF PROPOSED ALGORITHMS

The main advantage of proposed algorithms is that the datacentre which is used for processing takes less time when compared to the existing algorithms. It also reduces the average response time and can process the requests quickly.

CHAPTER 5

SYSTEM DESIGN AND IMPLEMENTATION

5.1 TECHNOLOGIES USED

This section contains a brief description of tools and technologies used in our algorithm.

5.1.1 JAVA

Java is an Object-Oriented programming language that operates with the help of objects. It supports the object-oriented features which are commonly used in present scenario. In cloud, these features help the developers to deploy and develop many applications and services.

Cloud computing provides a platform through which users can interact with the cloud services and continue to use them as needed. This platform must be accessible from any device whether it is a laptop, a mobile, a tablet or any other device.

Java has the ability to run on any platform that makes this possible. Java byte code made it very portable and secure for use. Many computational tasks performed in the cloud networks require smooth and quick networking services.

Java has many networking features which are used in the cloud computing. Java applet is one of the most popular features that revolutionized web [38] and internet technology.

5.1.2 CLOUDSIM

CloudSim [9] is a simulator that uses CloudSim Toolkit to simulate the cloud environment in surroundings. CloudSim is used to visualize cloud

infrastructure. CloudSim is used to simulate the cloud components. This means the connection of different types of clouds. The multi-layer architecture of CloudSim is shown in the Figure 5.1.

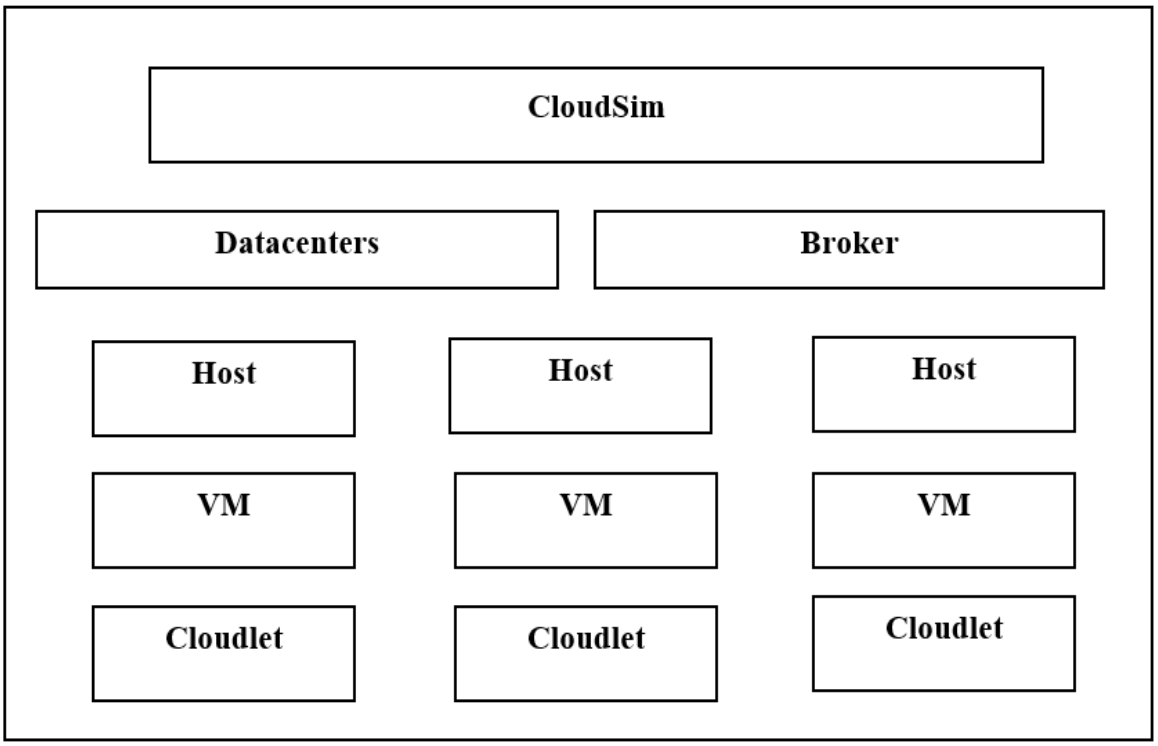


Figure 5.1 Architecture of cloudsim

5.1.3 CLOUD ANALYST

Cloud Analyst is a user-friendly environment. Cloud Analyst [38] is a simulator based on Graphical User Interface (GUI). It is used for modelling and analyzing applications. Cloud Analyst is built on the top of the CloudSim and it extends the properties of CloudSim. The components of Cloud Analyst are GUI Package, Region, User Base, Datacenter, DC Controller, Internet characteristics, Load Balancer, Server Broker.

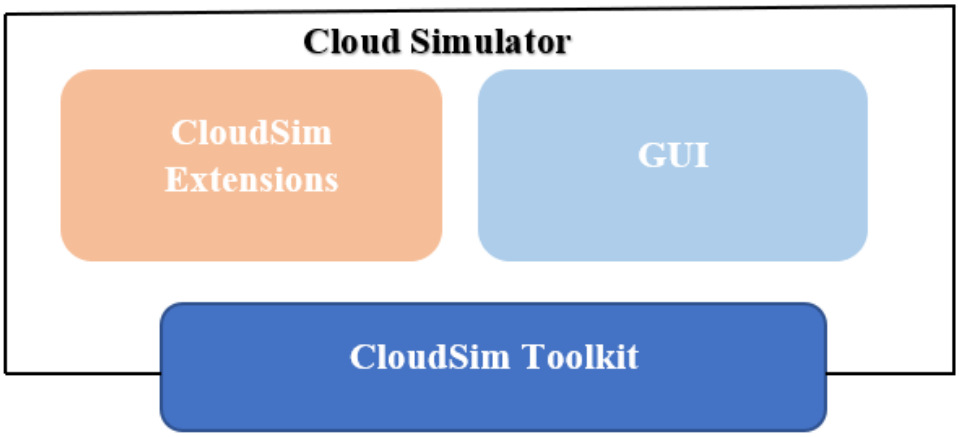


Figure 5.2 Components of cloudanalyst

5.2 SYSTEM REQUIREMENTS

The requirements for implementing the Hash ESCE algorithm is discussed in this section.

5.2.1 REGION ASSUMPTION

User Base and the Broker policies are located with the help of region. It is used for communication. R0, R1, R2, R3, R4 and R5. Table 5.1 shows the assumption of user based on the geographical location.

Table 5.1 Assumption of Regions

Region	Cloud analyst Region Id	Users
Africa	4	20
Oceania	5	30
Asia	3	80
North America	0	70
South America	1	50
Europe	2	50

5.2.2 USERBASE CONFIGURATION

In the userbase configuration the parameters like name, region, peak time start, request size, region, data size and off-peak time are mentioned. Table 5.2 displays the information of the user base and the data on each userbase.

Table 5.2 UserBase Parameters

User Base	Region	Time zone	Peak Hours (GMT)	Peak Hours (Local Time)	Online users simultaneously during peak hours	Online users during off peak hours
UB1	0	GMT -6.00	14.00-16.00	8.00-10.00 pm	500000	40000
UB2	1	GMT -4.00	16.00-18.00	8.00-10.00 pm	200000	20000
UB3	2	GMT -6.00	14.00-16.00	8.00-10.00 pm	400000	40000
UB4	3	GMT -6.00	14.00-16.00	8.00-10.00 pm	250000	25000
UB5	4	GMT -6.00	14.00-16.00	8.00-10.00 pm	150000	15000
UB6	5	GMT -6.00	14.00-16.00	8.00-10.00 pm	180000	18000

5.2.3 DATACENTER CONFIGURATION

Name, Region, Arch, Operating System (OS), Virtual Machine Manager (VMM), Cost, Memory, Storage, Data Transfer and Physical hardware are mentioned in the system.

5.3 COMPONENTS OF CLOUD ANALYST

Cloud analysts have many components for cloud modelling. They are,

- **GUI Package:** Java GUI
- **Region:** Datacentres and userbases are created in 6 regions.
- **Userbase:** Request to datacentre is sent from userbase
- **Datacentre:** It processes the requests from the userbase. It includes many VMs.
- **Datacentre Controller:** It is used to control the datacentres and VMs.
- **Internet Characteristics:** Characteristics of Bandwidth and latency are included here.
- **VM Load Balancer:** It is used to balance the incoming requests to the datacentre.
- **Service Broker:** It helps in clearing the traffic of requests in the web.

5.4 SYSTEM IMPLEMENTATION

The system implementation will be done in the following steps.

- Import **Cloud Analyst** in Eclipse
- **Datacentre Controller**

  DatacenterController.java in cloud analyst is used to add new algorithm which adds the algorithm in the cloud analyst workspace. Figure 5.3 shows the addition of Hash ESCE algorithm and Optimized algorithm in the *DatacenterController.java*.

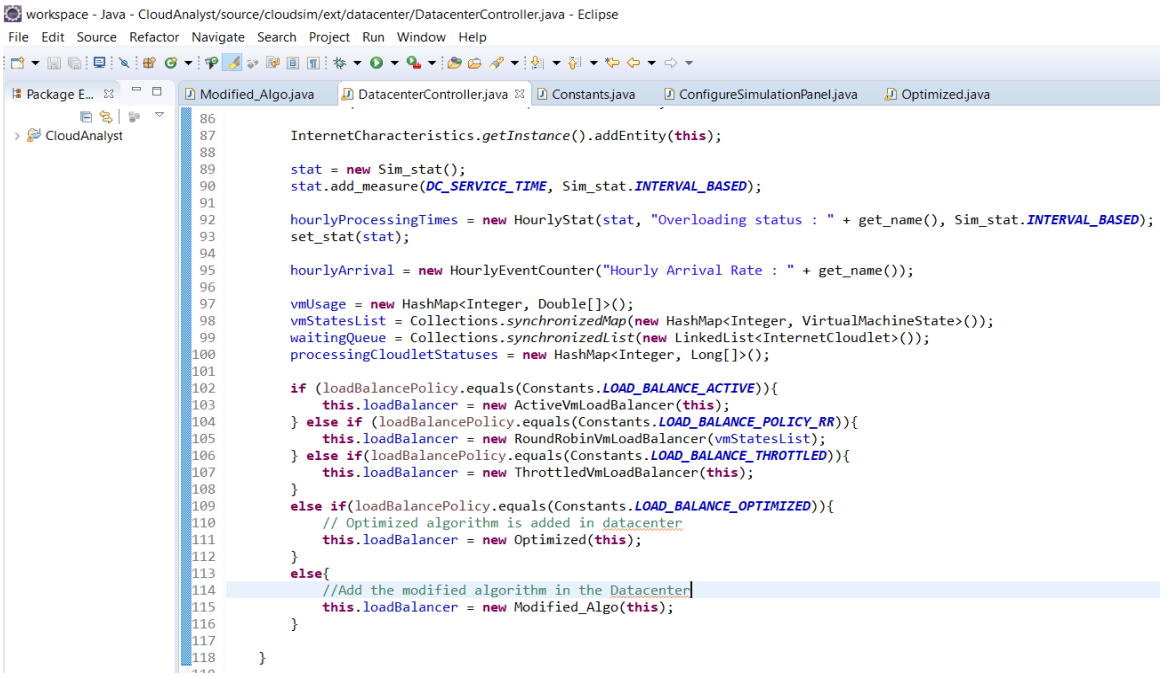


Figure 5.3 DatacenterController code snippet

❖ Adding Constants

The algorithm should have some constants to run the cloud analyst. Two constants *LOAD_BALANCED_MODIFIED* and *_OPTIMIZED* has been added in the *Constants.java* as shown in the Figure 5.4.

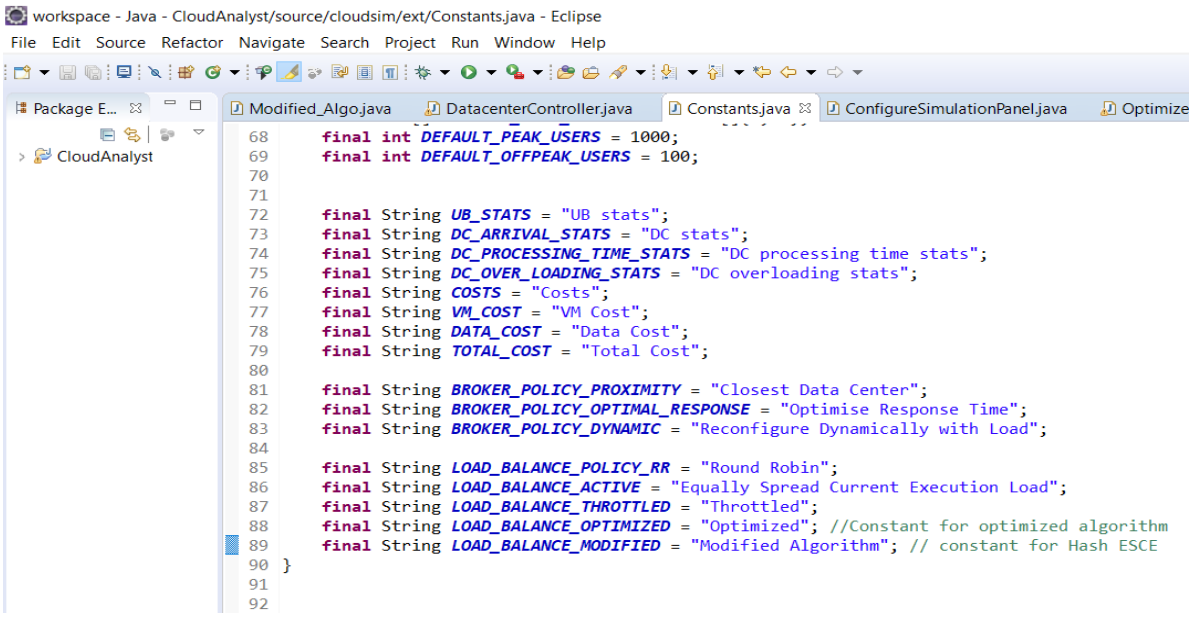


Figure 5.4 Constants.java code snippet

❖ **Configuration of Simulation Panel**

In the configuration panel, the Hash ESCE algorithm and the optimized algorithm has been called with its constants and it executes the program. In *ConfigureSimulationPanel.java*, implementation of our proposed algorithms is shown in the Figure 5.5.

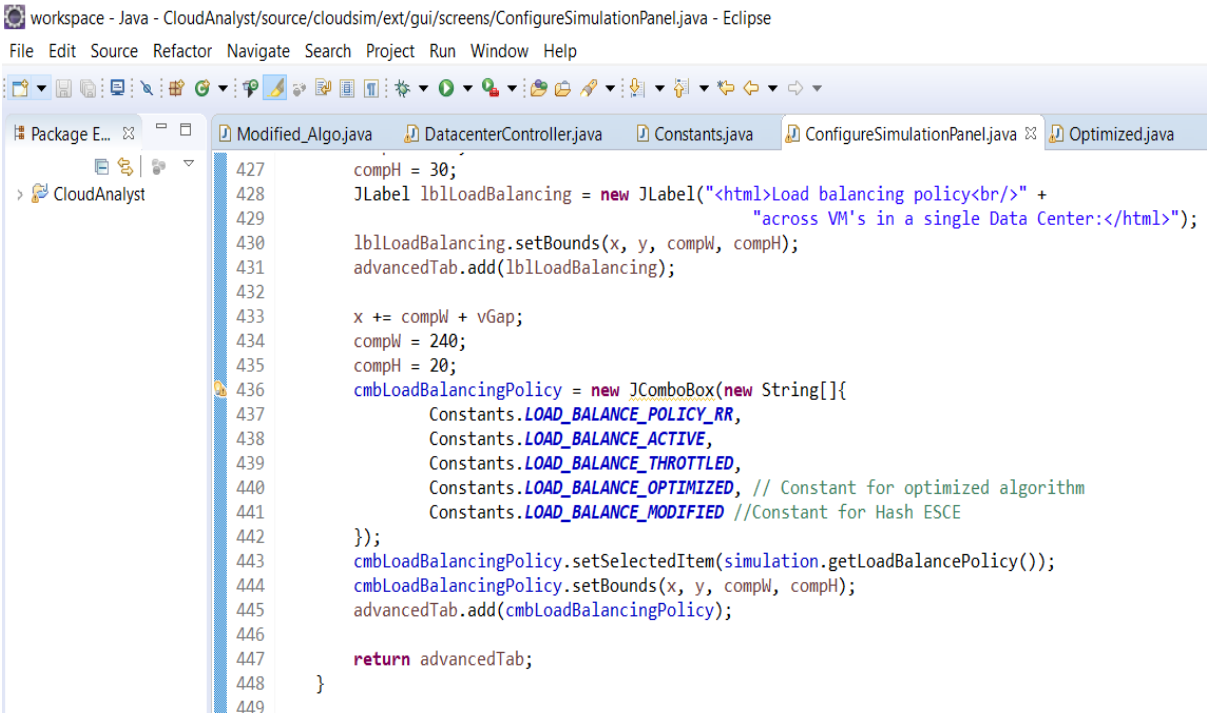


Figure 5.5 Code snippet of configuration simulation panel

❖ **Implementation of proposed algorithms**

In Eclipse, create a JAVA project and import the cloud analyst. In the cloud analyst, there is a package called *cloudsim.ext.datacenter*.

- **Implementation of Hash ESCE Algorithm**

Create new java class and implement the proposed algorithms in java. Implementation of Hash ESCE algorithm in the eclipse IDE using Java has been depicted in the Figure 5.6.

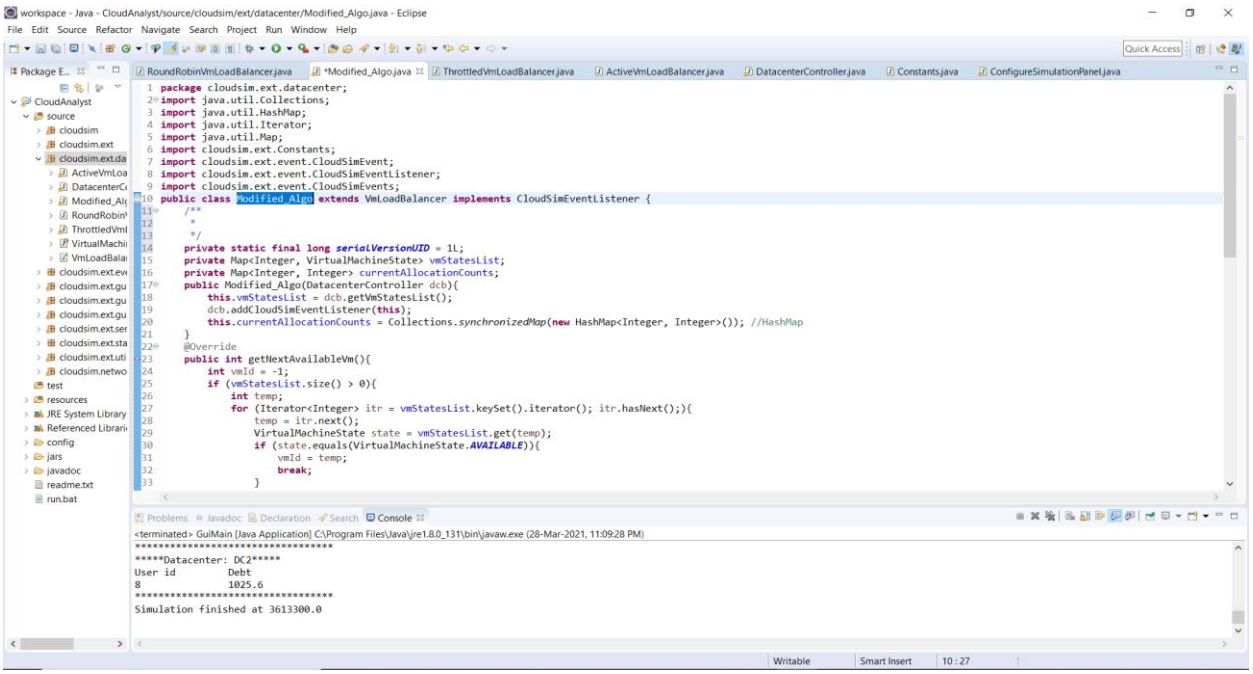


Figure 5.6 Implementation of hash ESCE algorithm in datacentre

- **Implementation of Optimized algorithm**

  Create new java class and implement the proposed algorithms in java. Implementation of Optimized algorithm in the eclipse IDE using Java has been depicted in the Figure 5.7.

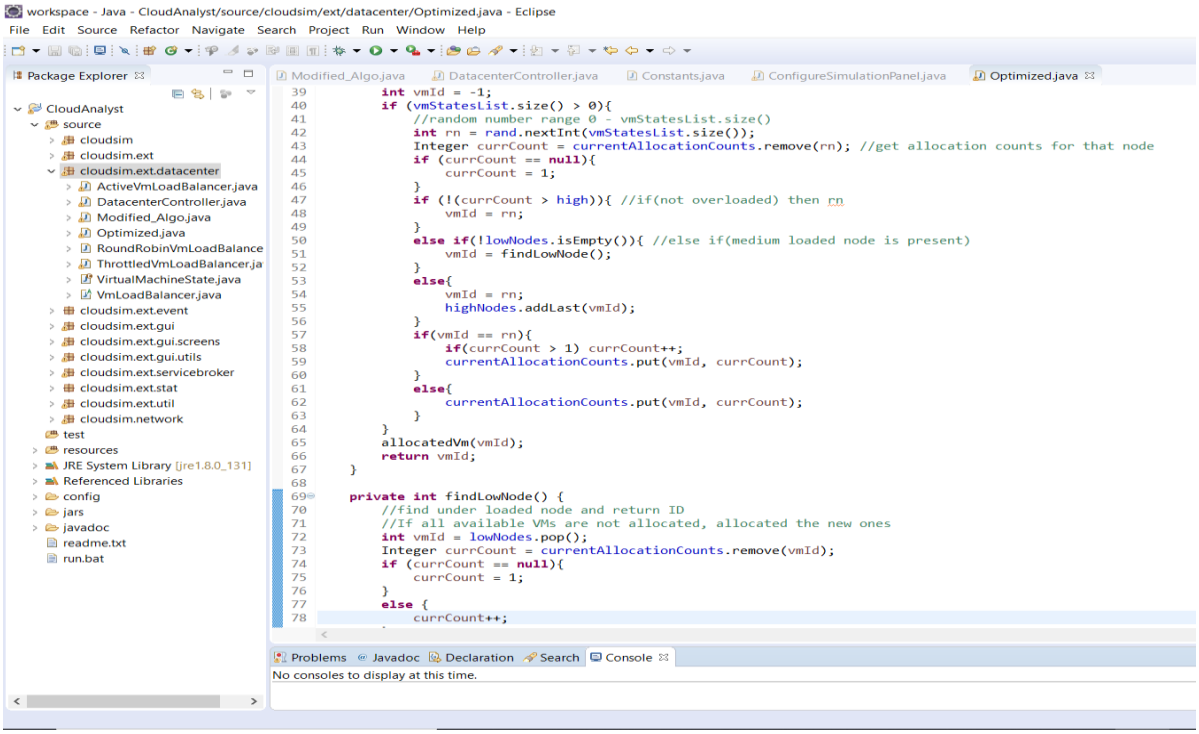


Figure 5.7 Implementation of optimized algorithm in datacentre

CHAPTER 6

RESULTS AND COMPARISON

Results are done in two ways. First, the modelling of the existing load balancing algorithm and Second, the modelling of the proposed algorithm. Then the results are compared with the existing algorithms in traditional approach. Graphical analysis of the results is done to get clear understanding of those approaches.

6.1 CLOUDANALYST WORKSPACE

This indicates the framework of Cloud Analyst Simulator. In this frame, work can assign the Userbases, Datacentres and different regions. The Cloud Analyst isn't having Graphical User Interface. Cloud Analyst uses Java Interface for simulation and it works in Eclipse Environment. The ranges are expressed as R0, R1, R2, R3, R4 and R5 as depicted in the Figure 6.1.

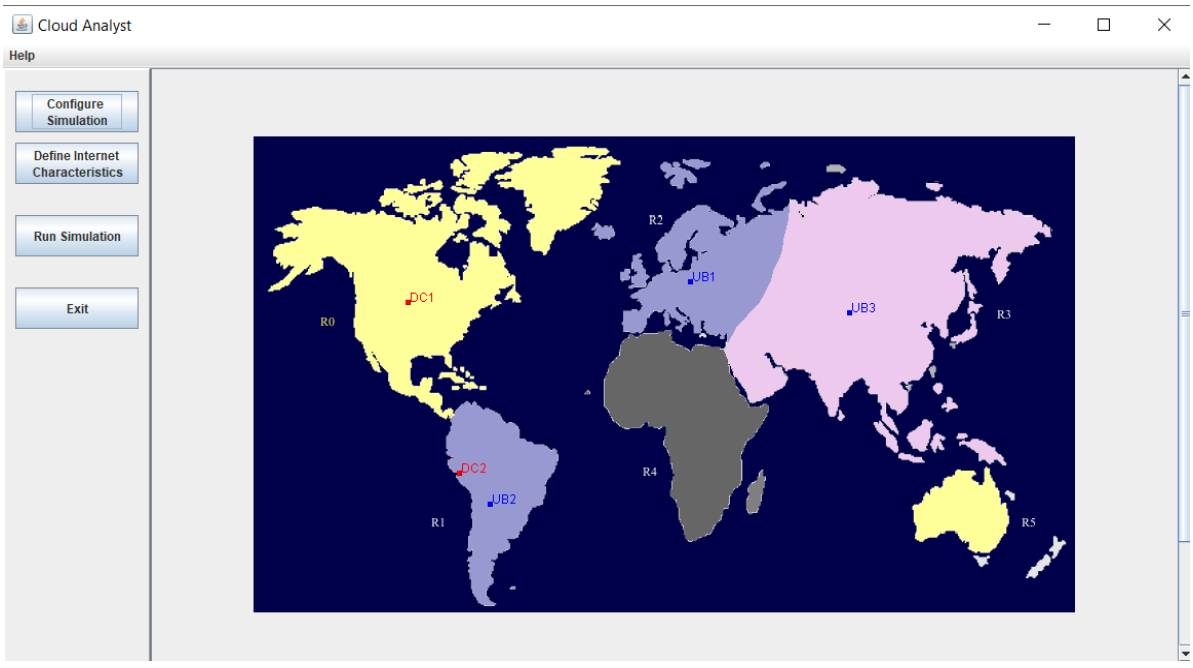


Figure 6.1 Cloud analyst workspace

6.2 SIMULATION RESULTS OF EXISTING ALGORITHMS

The overall response time and datacentre processing time can be analysed through the output. Generally, the average response time is considered for performance evaluation. It offers the minimal response time in comparison to others.

The simulation of ESCE, Round Robin and Throttled algorithm has been done in Cloud Analyst with two Datacentres and three userbases. For the Server Broker Policy, Closest Datacentre has been chosen which gives maximum profit to cloud services.

The Figure 6.2 shows the overall response time and datacentre processing time for the simulation of ESCE.

Overall Response Time Summary

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	281.57	40.37	615.06
Data Center processing time:	0.38	0.02	0.96

Figure 6.2 Overall time summary of ESCE algorithm

The simulation of Round Robin algorithm is done successfully and the overall time summary has been shown in the Figure 6.3.

Overall Response Time Summary

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	281.53	40.37	615.06
Data Center processing time:	0.38	0.02	0.96

Figure 6.3 Overall time summary of round robin algorithm

Overall response time and datacentre processing time summary of Throttled algorithm is shown in the Figure 6.4 after the simulation.

Overall Response Time Summary

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	281.41	40.32	615.06
Data Center processing time:	0.36	0.02	0.91

Figure 6.4 Overall time summary of throttled algorithm

Thus, the overall response time and datacentre processing time is taken from simulation of ESCE, RR and Throttled load balancing algorithms.

6.3 SIMULATION RESULTS OF PROPOSED ALGORITHMS

Average response time of Hash ESCE algorithm and Optimized algorithm have been analysed with the cloud analyst. The proposed Hash ESCE algorithm and Optimized algorithm has been implemented successfully and the simulation is done with same number of datacentres and userbases.

The Figure 6.5 shows the complete simulation of Hash ESCE algorithm with Closest Data Centre broker policy.

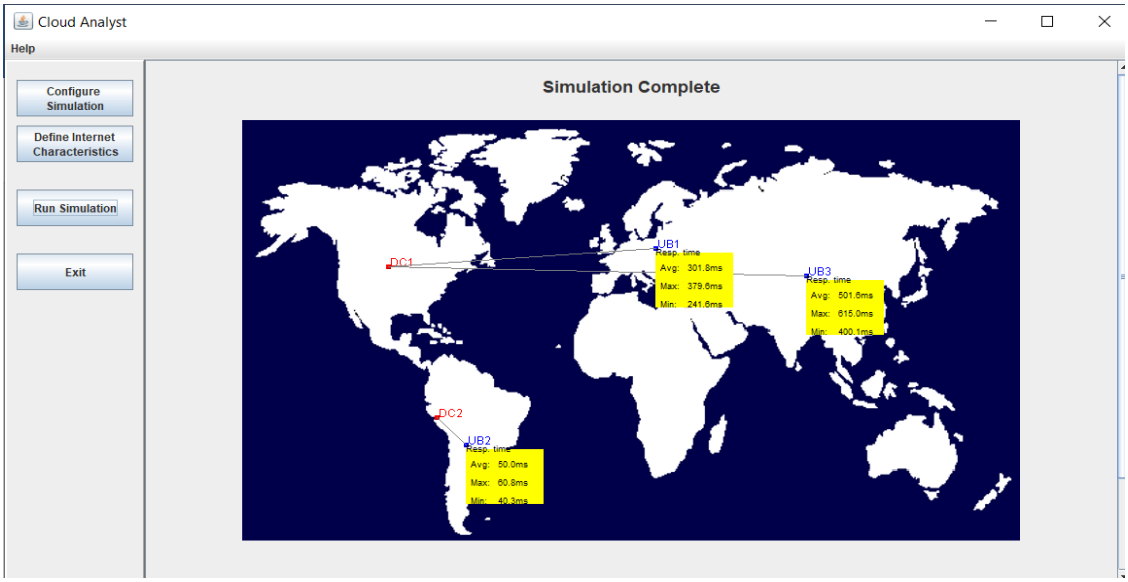


Figure 6.5 Simulation of hash ESCE algorithm

The complete response time summary after simulation of the Hash ESCE algorithm has been shown in the Figure 6.6.

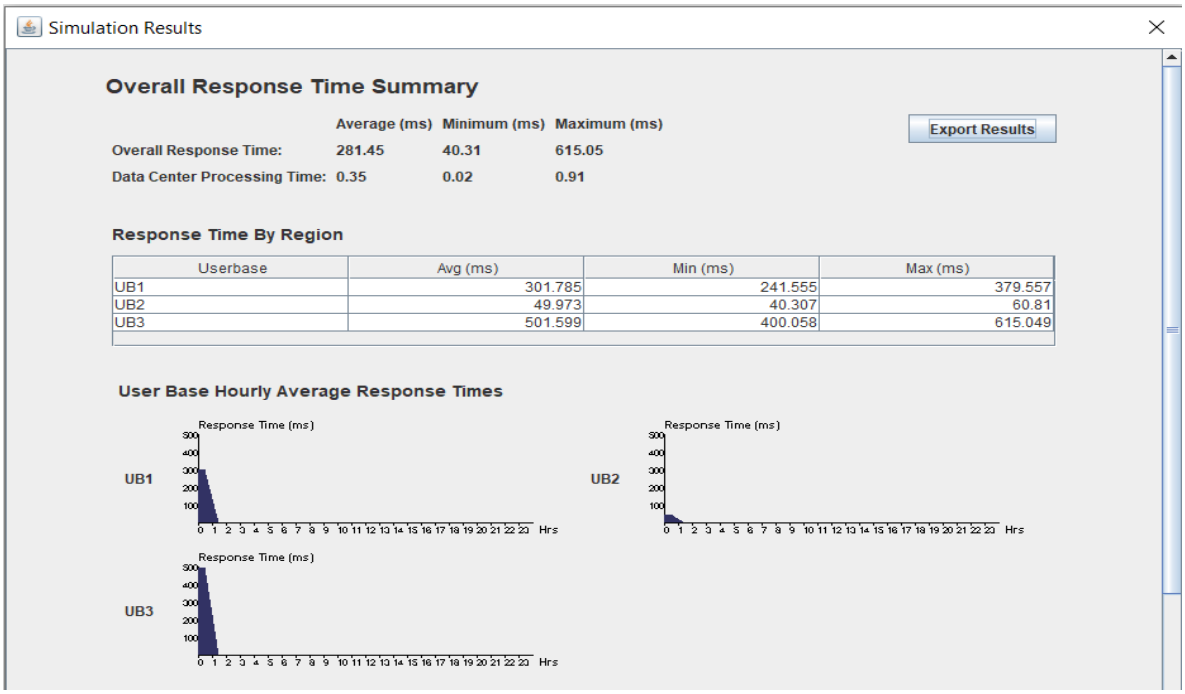


Figure 6.6 Response time summary of hash ESCE algorithm

The datacentre servicing time is obtained after the completion of simulation and cost is shown in the Figure 6.7.

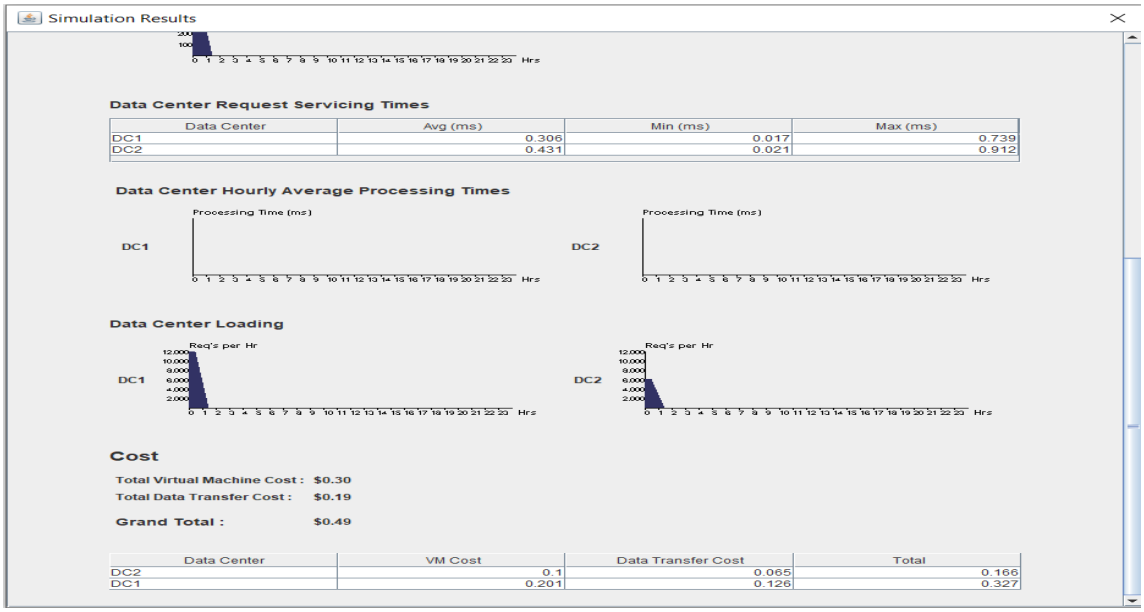


Figure 6.7 Datacentre processing time with cost of hash ESCE algorithm

The Figure 6.8 shows the complete simulation of optimized algorithm with Closest Data Centre broker policy.

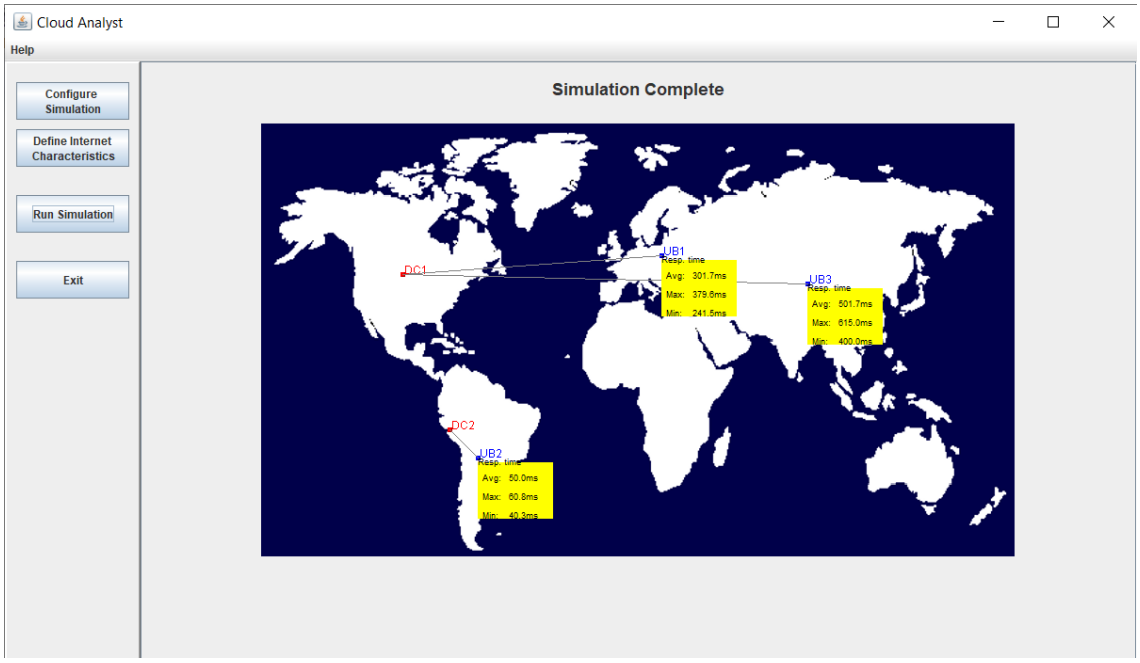


Figure 6.8 Simulation of optimized algorithm

The complete response time summary after simulation of the Optimized algorithm has been shown in the Figure 6.9.

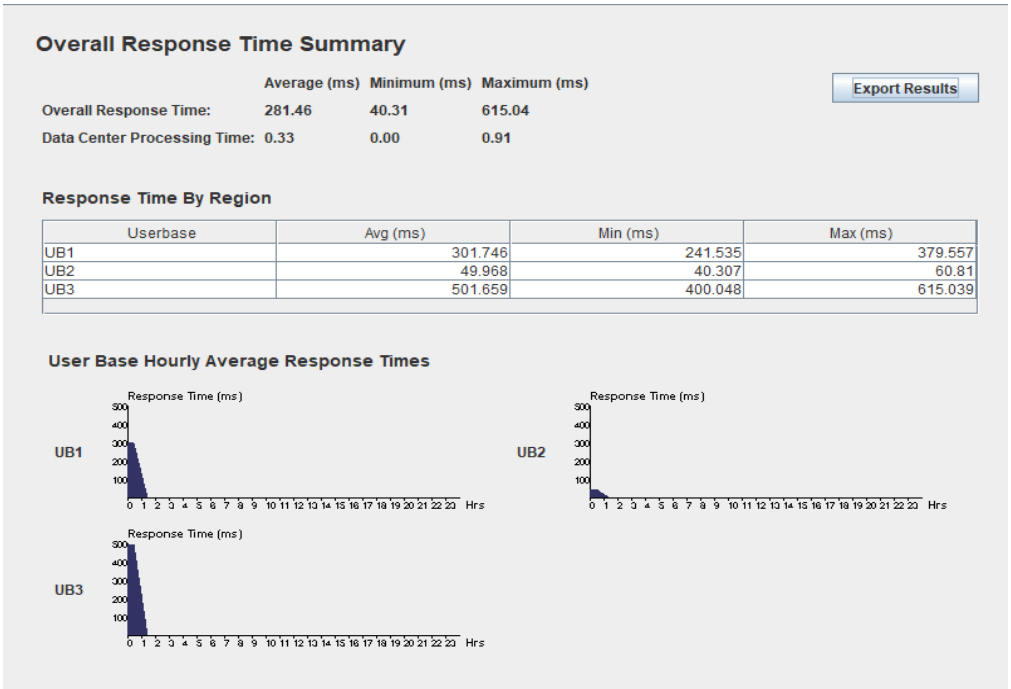


Figure 6.9 Response time summary of optimized algorithm

The datacentre servicing time is obtained after the completion of simulation and cost is shown in the Figure 6.10.

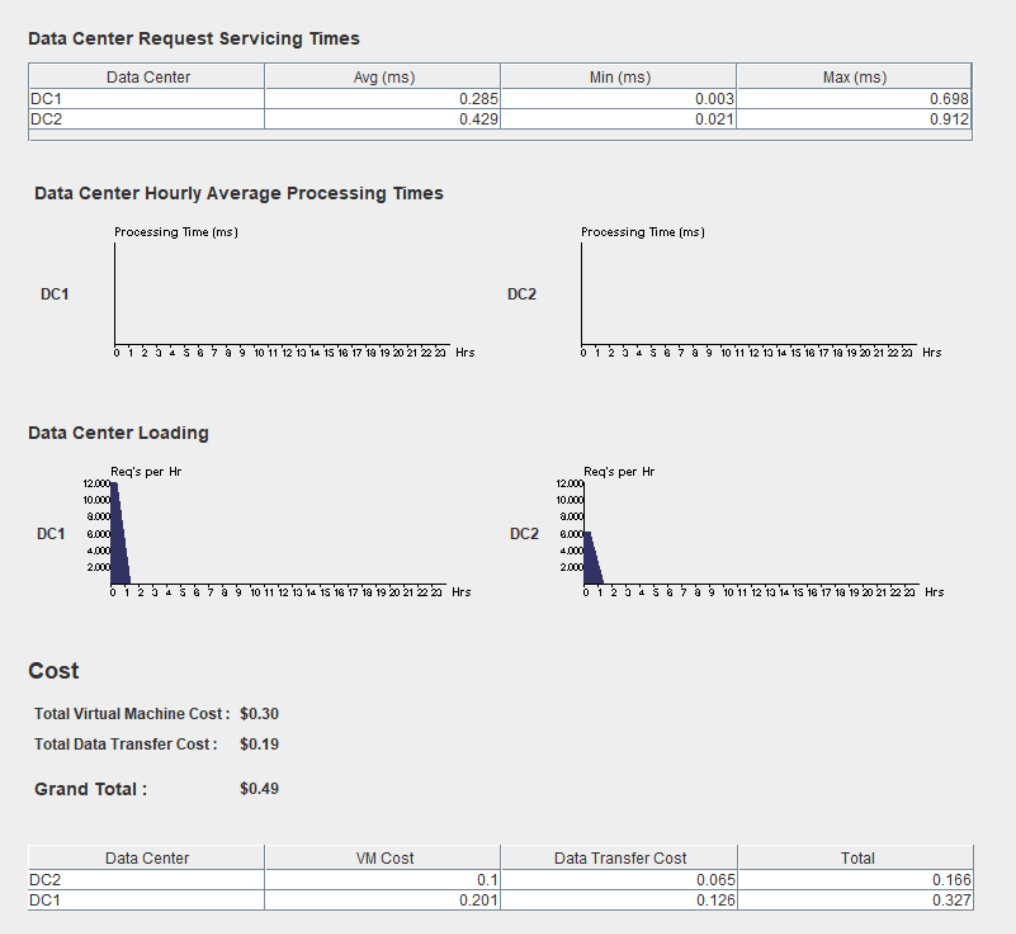


Figure 6.10 Datacentre processing time with cost of optimized algorithm

6.4 COMPARISON OF PROPOSED ALGORITHMS WITH EXISTING ALGORITHMS

The Hash ESCE algorithm and Optimized algorithm has been implemented successfully in the cloud analyst toolkit. The results of the proposed algorithms are compared with the existing algorithms.

The datacentre processing time and overall response time of VM is compared. Proposed algorithms give an output with less response time and less datacentre processing time.

The Figure 6.11 shows the comparison of the existing load balancing algorithm and proposed algorithms with two datacentres and three user bases.

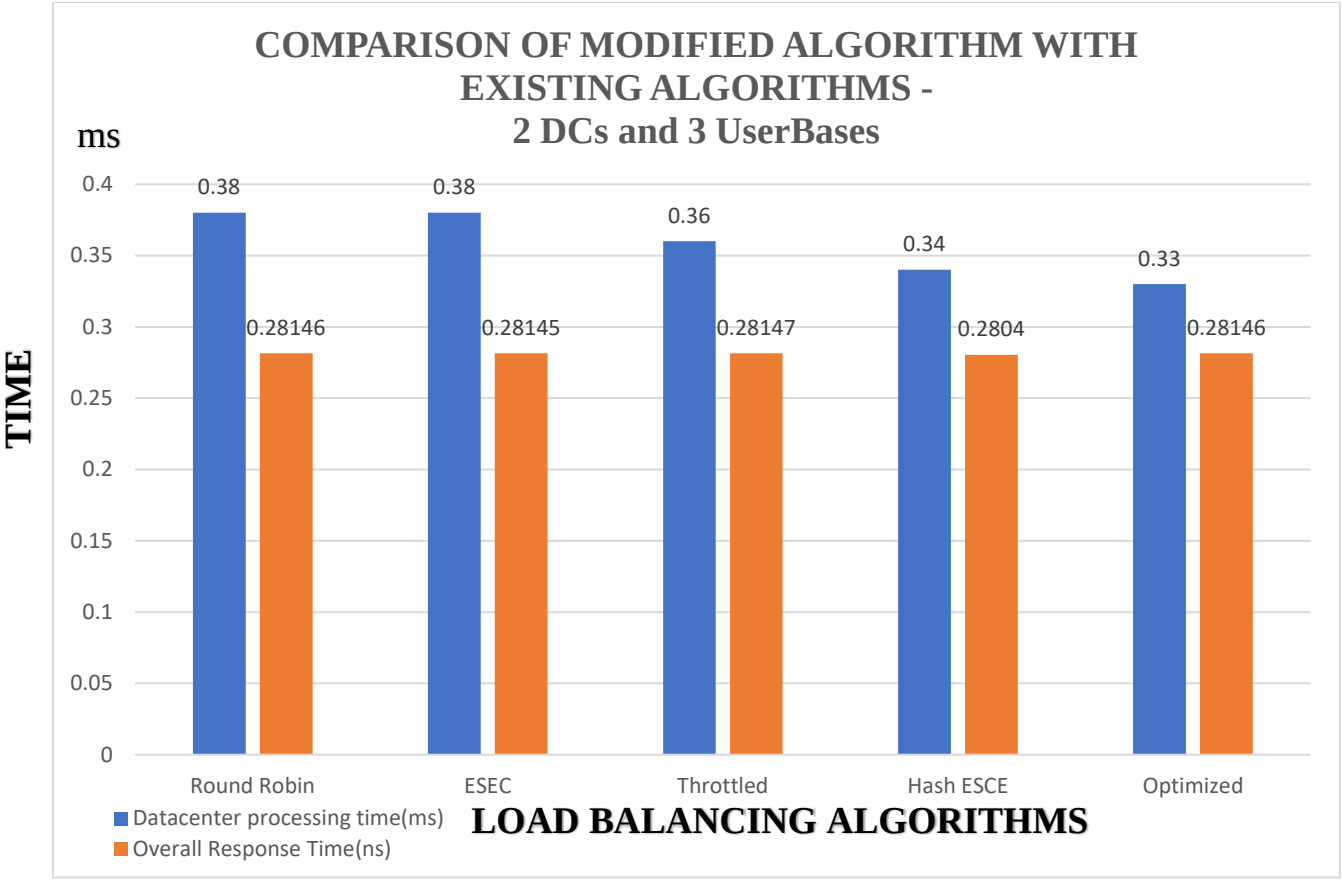


Figure 6.11 Comparison of algorithms with 2 DCs and 3 User Bases

The number of datacentres will also decide the response time and datacentre processing time in the cloud. That is, the number of datacenters is directly proportional to the data processing time and response time of the request. Thus, A comparison of the proposed algorithms and existing algorithms with 4 datacentres and 3 userbases is done and the outputs is plotted in the graph as shown in the Figure 6.12.

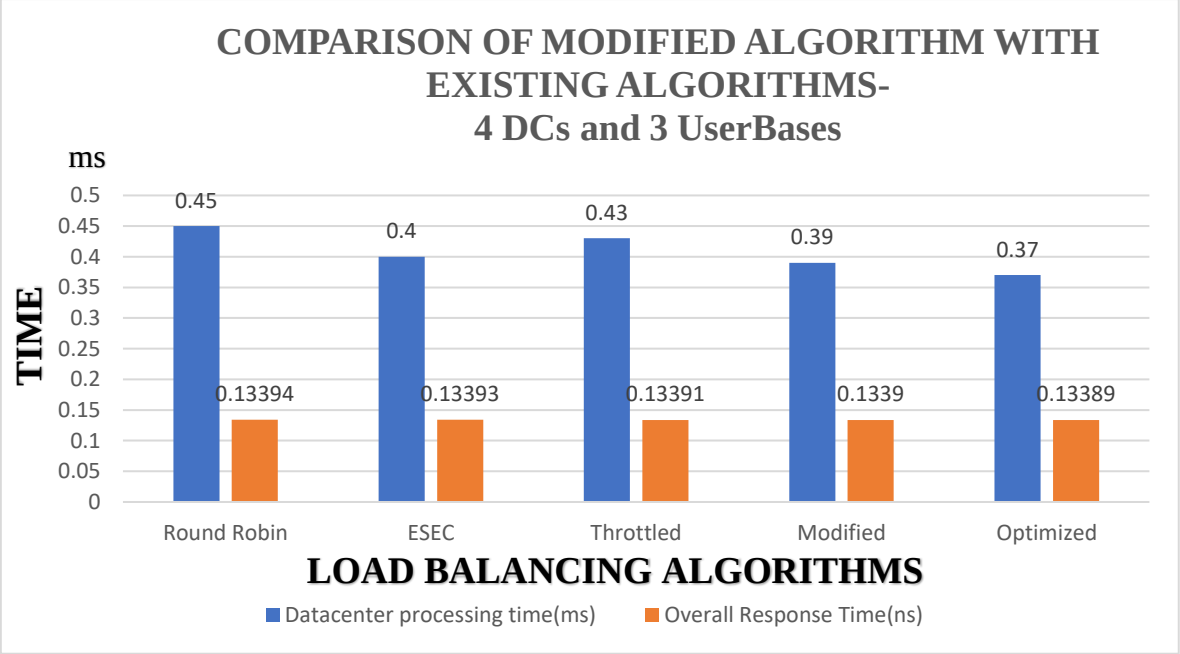


Figure 6.12 Comparison of algorithms with 4 DCs and 3 User Bases

Comparison of the modified algorithms and existing algorithms with 4 datacentres and 4 userbases is done. This comparison is done because the number of userbases will also change the response time and datacentre processing time. This comparison result is plotted as shown in the Figure 6.13.

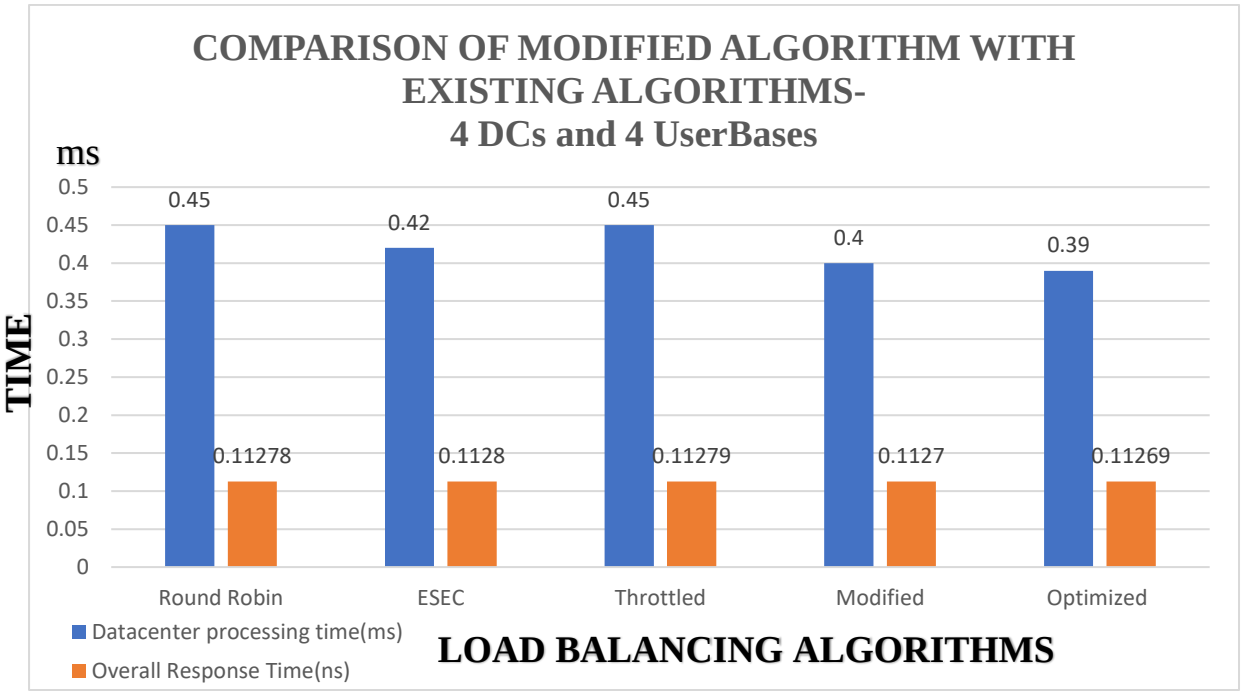


Figure 6.13 Comparison of algorithms with 4 DCs and 4 User Bases

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

The basic purpose of Hash ESCE algorithm and Optimized algorithms are to achieve less response time and datacentre processing time. Load balancing is a main part of the performance of the existing algorithms. The proposed Hash ESCE algorithm and Optimized algorithm is implemented in the cloud environment using Cloud Analyst. From the simulation results of the proposed Hash ESCE algorithm and Optimized algorithm, it is found that these algorithms take less datacentre processing time and response time. These algorithms work with no error in VM.

The load balancing algorithms are evaluated using the datacentre processing time and the response time. The response time and datacentre processing time are directly proportional to the number of datacenters and userbases. The project should design a load balancing policy which satisfy all the parameters which in turn increases the overall efficiency of the system.

In this project, analysis of parameters has been done in detail. The comparison of all the load balancing parameters are done which is used to evaluate the efficiency of the system. It also discusses various service broker policies in combination with various load balancing algorithms.

The service broker policy plays an important role in the real time traffic over network. In cloud computing system, all related operations are done in short period of time with higher efficiency.

7.2 FUTURE WORK

In future, a number of modifications can be done to get better results. The proposed method can be modified further to achieve power saving while making overall system energy efficient. It can also be made more time and cost efficient. These algorithms can be integrated into the lower layer. i.e., VM, where the VM is mapped to the cloudlets. This improves the efficiency of the system and shortens the overall response time. This work can be deployed in some real-time platforms to handle the real time traffic in the network and then evaluating the performance of the system.

APPENDIX

Code for Hash ESCE Algorithm:

```
package cloudsim.ext.datacenter;
import java.util.Collections;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import cloudsim.ext.Constants;
import cloudsim.ext.event.CloudSimEvent;
import cloudsim.ext.event.CloudSimEventListener;
import cloudsim.ext.event.CloudSimEvents;
public class Modified_Algo extends VmLoadBalancer implements
CloudSimEventListener {
        private Map<Integer, VirtualMachineState> vmStatesList;
        private Map<Integer, Integer> currentAllocationCounts;
         public Modified_Algo(DatacenterController dcb){
                this.vmStatesList = dcb.getVmStatesList();
                dcb.addCloudSimEventListener(this);
                this.currentAllocationCounts = Collections.synchronizedMap(new
HashMap<Integer, Integer>());
//HashMap - which is used to store the VMStates list with the status of those
VMStateslist
        }

@Override
        public int getNextAvailableVm(){
                int vmId = -1;
```

```
        if (vmStatesList.size() > 0){
                //Checks for the VMStatesList size
                int temp;
    for (Iterator<Integer> itr = vmStatesList.keySet().iterator();
    itr.hasNext();){
                temp = itr.next();
                VirtualMachineState state = vmStatesList.get(temp);
        if (state.equals(VirtualMachineState.AVAILABLE)){
                        // If the VM available then allocate VM
                                vmId = temp;
                                break;
                }else{
                //If the VM busy, then Load balancer checks for minimum
load
        if (currentAllocationCounts.size() < vmStatesList.size()){
                        // If hashmap size less than VMStates
                        if (!currentAllocationCounts.containsKey(temp)){
                                                        vmId = temp;
                                                        break;
    }} else {
        //If the Hashmap contains min load then the VM is allocated
        int currCount;
        int minCount = Integer.MAX_VALUE;
        for (int thisVmId : currentAllocationCounts.keySet()){
                currCount = currentAllocationCounts.get(thisVmId);
                if (currCount < minCount){
                        minCount = currCount;
                        vmId = thisVmId;
        }}}}}
```

```
                    allocatedVm(vmId);}
                    return vmId;
}
public void cloudSimEventFired(CloudSimEvent e) {
if (e.getId() ==CloudSimEvents.EVENT_CLOUDLET_ALLOCATED_TO_VM)
{
            int vmId = (Integer) e.getParameter(Constants.PARAM_VM_ID);
                    Integer currCount = currentAllocationCounts.remove(vmId)

if (currCount == null){
        currCount = 1;
} else {
        currCount++;}
currentAllocationCounts.put(vmId, currCount);
//vmStatesList.put(vmId, VirtualMachineState.BUSY);
}
else if (e.getId() == CloudSimEvents.EVENT_VM_FINISHED_CLOUDLET){
            int vmId = (Integer) e.getParameter(Constants.PARAM_VM_ID);
                    Integer currCount = currentAllocationCounts.remove(vmId);
                    if (currCount != null){
                            currCount--;
                            currentAllocationCounts.put(vmId, currCount);
            //vmStatesList.put(vmId, VirtualMachineState.AVAILABLE);
                    }}
}}
```

Code for Optimized Algorithm:

```
package cloudsim.ext.datacenter;
import java.util.ArrayDeque;
import java.util.Collections;
import java.util.Deque;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;
import cloudsim.ext.Constants;
import cloudsim.ext.event.CloudSimEvent;
import cloudsim.ext.event.CloudSimEventListener;
import cloudsim.ext.event.CloudSimEvents;
public class Optimized extends VmLoadBalancer implements
CloudSimEventListener {
	private Map<Integer, Integer> currentAllocationCounts;
	private Map<Integer, VirtualMachineState> vmStatesList;
	private final int low = 50, high = 150;
	private Deque<Integer> low Nodes;
	private Deque<Integer> highNodes;
	Random rand;
	public Optimized(DatacenterController dcb){
		dcb.addCloudSimEventListener(this);
		this.vmStatesList = dcb.getVmStatesList();
		rand = new Random();
		initializeUnderLoaded(); //initializing all to underLoaded nodes
		highNodes = new ArrayDeque<Integer>(vmStatesList.size());;
		this.currentAllocationCounts = Collections.synchronizedMap(new
HashMap<Integer, Integer>());
```

```
}
private void initializeUnderLoaded(){
        lowNodes = new ArrayDeque<Integer>(vmStatesList.size());
        for (int e : vmStatesList.keySet()){
                lowNodes.push(e);
}}
@Override
public int getNextAvailableVm(){
        int vmId = -1;
        if (vmStatesList.size() > 0){
                //random number range 0 - vmStatesList.size()
                int rn = rand.nextInt(vmStatesList.size());
                Integer currCount=currentAllocationCounts.remove(rn);
//get allocation counts for that node
                if (currCount == null){
                        currCount = 1;
                }if (!(currCount > high)){ //if(not overloaded) then rn
                        vmId = rn;
                }else if(!lowNodes.isEmpty()){
                //else if(medium loaded node is present)
                        vmId = findLowNode();
                }else{
                        vmId = rn;
                        highNodes.addLast(vmId);
                }if(vmId == rn){
                        if(currCount > 1) currCount++;
                        currentAllocationCounts.put(vmId, currCount);
                }else{
                        currentAllocationCounts.put(vmId, currCount);
```

```
                }}
            allocatedVm(vmId);
            return vmId;
        }
        private int findLowNode() {
            //find under loaded node and return ID
            //If all available VMs are not allocated, allocated the new ones
            int vmId = lowNodes.pop();
            Integer currCount = currentAllocationCounts.remove(vmId);
            if (currCount == null){
                currCount = 1;
            } else {
                currCount++;}
            if(currCount < low){
                lowNodes.addLast(vmId);}
            currentAllocationCounts.put(vmId, currCount);
            return vmId;
        }
        public void cloudSimEventFired(CloudSimEvent e) {
            if (e.getId() ==
CloudSimEvents.EVENT_CLOUDLET_ALLOCATED_TO_VM){
                int vmId = (Integer)
e.getParameter(Constants.PARAM_VM_ID);
                Integer currCount = currentAllocationCounts.remove(vmId);
                if (currCount == null){
                    currCount = 1;}
                else {
                    currCount++;}
                currentAllocationCounts.put(vmId, currCount);
```

```
                } else if (e.getId() ==
CloudSimEvents.EVENT_VM_FINISHED_CLOUDLET){
                        int vmId = (Integer)
e.getParameter(Constants.PARAM_VM_ID);
                        Integer currCount = currentAllocationCounts.remove(vmId);
                        if (currCount != null){
                                currCount--;
                                currentAllocationCounts.put(vmId, currCount);
                                if(currCount < low){
                                        lowNodes.addLast(vmId);//check VM for under
and over load
                                }}
                        if (currCount == null){
                                lowNodes.push(vmId);
                        }}
}}
```

Datacentre Controller.java

```
package cloudsim.ext.datacenter;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.HashMap;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;
```

```
import cloudsim.DataCenter;
import cloudsim.DatacenterBroker;
import cloudsim.DatacenterCharacteristics;
import cloudsim.DatacenterTags;
import cloudsim.VMCharacteristics;
import cloudsim.VirtualMachine;
import cloudsim.ext.Constants;
import cloudsim.ext.GeoLocatable;
import cloudsim.ext.InternetCharacteristics;
import cloudsim.ext.InternetCloudlet;
import cloudsim.ext.event.CloudSimEvent;
import cloudsim.ext.event.CloudSimEventListener;
import cloudsim.ext.event.CloudSimEvents;
import cloudsim.ext.event.CloudsimObservable;
import cloudsim.ext.stat.HourlyEventCounter;
import cloudsim.ext.stat.HourlyStat;
import cloudsim.ext.util.CommPath;
import cloudsim.ext.util.InternetEntitityRegistry;
import eduni.simjava.Sim_event;
import eduni.simjava.Sim_stat;
import eduni.simjava.Sim_system;
import gridsim.GridSim;
```

```
import gridsim.GridSimTags;

public class DatacenterController extends DatacenterBroker implements GeoLocatable, CloudsimObservable, Constants {

    private List<CloudSimEventListener> listeners;

    private VmLoadBalancer loadBalancer;

    private int region;

    private Sim_stat stat;

    private int queuedCount = 0;

    private double costPerVmHour;

    private double costPerDataGB;

    private double totalData;

    private HourlyEventCounter hourlyArrival;

    private HourlyStat hourlyProcessingTimes;

    private Map<Integer, Double[]> vmUsage;

    private Map<Integer, VirtualMachineState> vmStatesList;

    private Map<Integer, Long[]> processingCloudletStatuses;

    private int requestsPerCloudlet;

    private List<InternetCloudlet> waitingQueue;

    private String dcName;

    private boolean lastVmCreateFailed = false;

    private int allRequestsProcessed = 0;
```

```
public DatacenterController (String name, int region, double costPerVmHour,
double costPerDataGB, int requestsPerCloudlet, String loadBalancePolicy) throws Exception {
        super(name + "-Broker");
        this.dcName = name;
        System.out.println("Creating new broker " + get_name());
        listeners =new ArrayList<CloudSimEventListener>();
        this.region = region;
        this.costPerVmHour = costPerVmHour;
        this.costPerDataGB = costPerDataGB;
        this.requestsPerCloudlet = requestsPerCloudlet;
        InternetCharacteristics.getInstance().addEntity(this);
        stat = new Sim_stat();
        stat.add_measure(DC_SERVICE_TIME,
Sim_stat.INTERVAL_BASED);
        hourlyProcessingTimes = new HourlyStat(stat, "Overloading status
: " + get_name(), Sim_stat.INTERVAL_BASED);
        set_stat(stat);
    hourlyArrival = new HourlyEventCounter("Hourly Arrival Rate : " +
get_name());
    vmUsage = new HashMap<Integer, Double[]>();
```

```
vmStatesList = Collections.synchronizedMap (new HashMap <Integer, VirtualMachineState> ());

waitingQueue = Collections.synchronizedList (new LinkedList <InternetCloudlet>());

processingCloudletStatuses = new HashMap<Integer, Long[]>();

if (loadBalancePolicy.equals(Constants.LOAD_BALANCE_ACTIVE)){

        this.loadBalancer = new ActiveVmLoadBalancer(this);

    }else if

(loadBalancePolicy.equals (Constants. LOAD_BALANCE_POLICY_RR))

{   this.loadBalancer = new RoundRobinVmLoadBalancer(vmStatesList);

}else if

(loadBalancePolicy.equals(Constants.LOAD_BALANCE_THROTTLED)){

        this.loadBalancer = new ThrottledVmLoadBalancer(this);

}else if

(loadBalancePolicy.equals(Constants.LOAD_BALANCE_OPTIMIZED)){

        this.loadBalancer = new Optimized(this); // Add optimized algorithm in datacenter}

else { this.loadBalancer = new Modified_Algo(this); //Add the modified algorithm in the Datacenter

}}
```

Constants.java

```
package cloudsim.ext;
public interface Constants {
	final String STANDARD_SEPARATOR = "-";
	final int WORLD_REGIONS = 6;
	final String INTERNET = "Internet";
	final int REQUEST_INTERNET_CLOUDLET_TAG = 2001;
	final int RESPONSE_INTERNET_CLOUDLET_TAG = 2002;
	final String MEASURE_TYPE_OVERALL_USER_BASE_RESPONSE
= "Overall userbase response time";
	final String MEASURE_TYPE_USER_BASE_RESPONSE = "Userbase
Response Time";
	final String MEASURE_TYPE_DC_PROCESSING_TIME = "DC
Processing Time";
	final String PARAM_PROCESSING_TIME = "processing_time";
	final String PARAM_COMM_PATH = "commPath";
	final double MILLI_SECONDS_TO_DAYS = 1000 * 60 * 60 * 24;
	final int DEFAULT_APP_ID = 1;
	final String SIMULATION_COMPLETED_TIME =
"sim_completed_at";
	final String PDF_EXTENSION = ".pdf";
	//Default Data Center characteristics
	final String DEFAULT_DATA_CENTER_NAME = "DC1";
	final String DEFAULT_ARCHITECTURE = "x86";
	final String DEFAULT_OS = "Linux";
	final String DEFAULT_VMM = "Xen";
	final int DEFAULT_DC_REGION = 0;
	final int DEFAULT_VM_COUNT = 5;
```

```
final double DEFAULT_COST_PER_PROC = 0.1; // the cost of using processing in this resource
final double DEFAULT_COST_PER_MEM = 0.05; // the cost of using memory in this resource
final double DEFAULT_COST_PER_STOR = 0.1; // the cost of using storage in this resource
final double DEFAULT_COST_PER_BW = 0.1;
final int DEFAULT_MC_MEMORY = 204800;
final long DEFAULT_MC_STORAGE = 100000000;
final int DEFAULT_MC_BW = 1000000;
final int DEFAULT_MC_PROCESSORS =  4;
final int DEFAULT_MC_SPEED = 10000;
final long DEFAULT_VM_IMAGE_SIZE = 10000;//MB
final int DEFAULT_VM_MEMORY = 512;//MB
final long DEFAULT_VM_BW = 1000;
//Default User base characteristics
final String DEFAULT_USER_BASE_NAME = "UB1";
final int DEFAULT_UB_REGION = 2;
final int DEFAULT_REQ_PER_USER_PER_HR = 60;
final long DEFAULT_REQ_SIZE = 100;
final int[] DEFAULT_PEAK_HOURS = new int[]{3, 9};
final int DEFAULT_PEAK_USERS = 1000;
final int DEFAULT_OFFPEAK_USERS = 100;
final String UB_STATS = "UB stats";
final String DC_ARRIVAL_STATS = "DC stats";
final String DC_PROCESSING_TIME_STATS = "DC processing time stats";
final String DC_OVER_LOADING_STATS = "DC overloading stats";
final String COSTS = "Costs";
```

```
        final String VM_COST = "VM Cost";
        final String DATA_COST = "Data Cost";
        final String TOTAL_COST = "Total Cost";
        final String BROKER_POLICY_PROXIMITY = "Closest Data Center";
        final String BROKER_POLICY_OPTIMAL_RESPONSE = "Optimise
Response Time";
        final String BROKER_POLICY_DYNAMIC = "Reconfigure
Dynamically with Load";
        final String LOAD_BALANCE_POLICY_RR = "Round Robin";
        final String LOAD_BALANCE_ACTIVE = "Equally Spread Current
Execution Load";
        final String LOAD_BALANCE_THROTTLED = "Throttled";
        final String LOAD_BALANCE_OPTIMIZED = "Optimized";
        final String LOAD_BALANCE_MODIFIED = "Modified Algorithm";}
```

ConfigureSimulationPanel.java

```
private JPanel createAdvancedTab(){
                int leftMargin = 50;
                int x = leftMargin;
                int y = 50;
                int vGap = 20;
                JPanel advancedTab = new JPanel();
                advancedTab.setLayout(null);
                int compW = 500;
                int compH = 20;
```

```
compW = 240;

int lastCompH = compH = 60;

JLabel lblUserGroup = new JLabel("<html>User grouping factor in User Bases:" +"<br/>(Equivalent to number of simultaneous" + "<br/> users from a single user base)</html>");

lblUserGroup.setBounds(x, y, compW, compH);

advancedTab.add(lblUserGroup);

x += compW + vGap;

y += 10;

compW = 80;

compH = 20;

txtUserGroupingFactor = new JTextField ("" + simulation. getUserGroupingFactor());

txtUserGroupingFactor.setBounds(x, y, compW, compH);

advancedTab.add(txtUserGroupingFactor);

x = leftMargin;

y += lastCompH + vGap;

compW = 240;

lastCompH = compH = 70;

JLabel lblDcRequestGrouping = new JLabel("<html>Request grouping factor in Data Centres:" + "<br/>(Equivalent to number of simultaneous" + "<br/> requests a single applicaiton server" + "<br/> instance can support.) </html>");
```

```
lblDcRequestGrouping.setBounds(x, y, compW, compH);

advancedTab.add(lblDcRequestGrouping);

x = leftMargin;

y += lastCompH + vGap;

compW = 240;

compH = 30;

JLabel lblInstructionLength = new JLabel("<html>Executable instruction length per request:" + "<br/>(bytes)</html>");

lblInstructionLength.setBounds(x, y, compW, compH);

advancedTab.add(lblInstructionLength);

x += compW + vGap;

compW = 80;

compH = 20;

txtInstructionLength = new JTextField("" + simulation.getInstructionLengthPerRequest());

txtInstructionLength.setBounds(x, y, compW, compH);

advancedTab.add(txtInstructionLength);

x = leftMargin;

y += lastCompH + vGap;

compW = 240;

compH = 30;
```

```
JLabel lblLoadBalancing = new JLabel("<html>Load balancing policy<br/>" +"across VM's in a single Data Center:</html>");
lblLoadBalancing.setBounds(x, y, compW, compH);
advancedTab.add(lblLoadBalancing);
x += compW + vGap;
compW = 240;
compH = 20;
cmbLoadBalancingPolicy = new JComboBox(new String[]{
        Constants.LOAD_BALANCE_POLICY_RR,
        Constants.LOAD_BALANCE_ACTIVE,
        Constants.LOAD_BALANCE_THROTTLED,
        Constants.LOAD_BALANCE_OPTIMIZED,
        // Add Optimized constant for optimized algorithm
        Constants.LOAD_BALANCE_MODIFIED
        //Add a constant for Hash ESCE
});
cmbLoadBalancingPolicy.setSelectedItem(simulation.getLoadBalancePolicy());
cmbLoadBalancingPolicy.setBounds(x, y, compW, compH);
advancedTab.add(cmbLoadBalancingPolicy);
return advancedTab;
}
```

REFERENCES

[1] Ajay Gulati, Ranjeev.K.Chopra, "Dynamic Round Robin for Load Balancing in a Cloud Computing", IJCSMC, June 2013.

[2] Alexandru Iosup, Member, IEEE, Simon Ostermann,Nezih Yigitbasi, Member, IEEE, Radu Prodan, Member, IEEE, Thomas Fahringer, Member, IEEE, and Dick Epema, Member, IEEE, "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing", IEEE TPDS, MANY-TASK COMPUTING, NOVEMBER 2010.

[3] Ankush P.Deshmukh and Prof. Kumarswamy Pamu "Applying Load Balancing: A Dynamic Approach" (IJARCSSE), vol. 2, issue 6, June 2012.

[4] Anthony T.Velte, Toby J.Velte, Robert Elsenpeter, Cloud Computing A Practical Approach, TATA McGRAWHILL Edition 2010.

[5] Bhathiya Wickremasinghe, "CloudAnalyst: A CloudSim based Tool for Modeling Chapter7 Page 69 and Analysis of Large-Scale Cloud Computing Environments" MEDC project report, 433-659 Distributed Computing project, CSSE department., University of Melbourne, 2009.

[6] Che-Lun Hung, Hsiao-hsi Wang and Yu-Chen Hu, "Efficient Load Balancing Algorithm for Cloud Computing Network," Dept. of Computer Science & Communication Engineering, Providence University 200 Chung Chi Rd., Taichung 43301, Republic of China (Taiwan).

[7] Ching-Chi Lin, Pangfeng Liu, Jan-Jan Wu, "Energy-Aware Virtual Machine Dynamic Provision and Scheduling for Cloud Computing", IEEE 4th International Conference on Cloud Computing, 2011.

[8] D A Menasce, P NGO, "Understanding Cloud Computing: Experimentation and Capacity Planning", Proc. Computer Measurement Group Conf, Dallas, TX, Dec. 7-11, 2009.

[9] Definitions of cloud simulator: http://www.cloudbus.org/cloudsim

[10] Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360- degree compared. In Proceedings of Grid Computing Environments Workshop, pages 1–10, 2008.

[11] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. Communications of the ACM, 51(1):107–113, 2008.

[12] Jaspreet kaur "Comparison of Load balancing algorithms in a Cloud" International Journal of Engineering Research and Applications" (IJERA), vol. 2, issue 3, May, June 2012.

[13] Jeffrey M. Galloway, Karl L. Smith, Susan S. Vrbsky, "Power Aware Load Balancing for Cloud Computing", Proceedings of the World Congress on Engineering and Computer Science 2011 Vol I WCECS 2011, October 19-21, 2011.

[14] Jingnan Yao Jiani Guo; Bhuyan, L.N., "Ordered Round-Robin: An Efficient Sequence Preserving Packet Scheduler" IEEE transactions, vol. 57, issue: 12, 30 May, 2008.

[15] Kumar Nishant, Pratik Sharma, Vishal Krishna, Chhavi Gupta and Kuwar Pratap Singh, Nitin and Ravi Rastogi, “Load Balancing of Nodes in Cloud Using Ant Colony Optimization” IEEE 2012 14th International Conference on Modelling and Simulation.

[16] M. Armbrust A. Fox, and R. Griffith. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[17] Md. S. Q. Zulkar Nine, Md. Abul Kalam Azad, Saad Abdullah, Rashedur M Rahman, “Fuzzy Logic Based Dynamic Load Balancing in Virtualized Data Centres”, Fuzzy Systems (FUZZ), 2013 IEEE International Conference.

[18] Milan E. Soklic “Simulation of Load balancing algorithms” ACM - SIGCSE Bulletin, December, 2002.

[19] Mishra, Ratan, Jaiswal, Anant,P “Ant Colony Optimization: A Solution Of Load Balancing In Cloud”, April 2012, International Journal Of Web & Semantic Technology;Apr2012, Vol. 3 Issue 2, P33

[20] Nidhi Jain Kansal and Inderveer Chana, “Existing Load Balancing Techniques in Cloud Computing: A systematic Review”,Journal of Information Systems and Communication, 2012.

[21] P. Mell and T. Grance. The NIST Definition of Cloud Computing (Draft). National Institute of Standards and Technology, 53:7, 2010.

[22] Panagiotis Kalagiakos, Panagiotis Karampelas, "Cloud Computing Learning" in the Proceeding of IEEE International Conference on Application of Information and Communication Technologies, Baku,Oct. 2011.

[23] Patrick Naughton and Herbert Schildt, Complete Reference Osborne/McGraw-Hill 1999.

[24] R. Buyya, R. Ranjan, and R. N. Calheiros, “Modeling And Simulation Of Scalable Cloud Computing Environments And The Cloudsim Toolkit: Challenges And Opportunities,” Proc. Of The 7th High Performance Computing and Simulation Conference (HPCS 09), IEEE Computer Society, June 2009.

[25] Radojevic, B. & Zagar, M. (2011). Analysis of issues with load balancing algorithms in hosted (cloud) environments. In proceedings of 34th International Convention on MIPRO, IEEE.

[26] Randles, M., D. Lamb and A. Taleb-Bendiab, “A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing,” in Proc. IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Perth, Australia, April 2010.

[27] Raul´ Alonso-Calvo, Jose Crespo, Miguel Garc´ıa-Remesal, Alberto Anguita and Victor Maojo, “On distributing load in cloud computing: A real application for very-large image datasets”, International Conference on Computational Science, ICCS 2010, pp.-2669- 2677, 2010.

[28] Shridhar G.Domanal and G.Ram Mohana Reddy "Load Balancing in Cloud Computing Using Modified Throttled Algorithm" IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), October 2013.

[29] Shu-Ching Wang, Kuo-Qin Yan, Wen-Pin Liao, Shun-Sheng Wang, "Towards a Load Balancing in a Three-level Cloud Computing Network", 2010 IEEE, pp. 108-113.

[30] Soumya Ray and Ajanta De Sarkar "Execution Analysis of Load Balancing Algorithms in Cloud Computing Environment" (IJCCSA), vol.2, no.5, October 2012.

[31] Srinivas Sethi, Anupama Sahu, Suvendu Kumar Jena, "Efficient load Balancing in Cloud Computing using Fuzzy Logic", IOSRJEN July 2012.

[32] Subasish Mohapatra, Subhadarshini Mohanty, K.Smruti Rekha, "Analysis of Different Variants in Round Robin Algorithms for Load Balancing in Cloud Computing", IJCA, May 2013.

[33] Sun Microsystems, Inc."Introduction to Cloud Computing Architecture" Whitepaper, Ist Edition, June 2009.

[34] T V R Anandarajan, M A Bhagyabini, "Co-operative scheduled Energy aware load balancing technique for an efficient computational cloud", IJCSI, volume 8, issue March, 2011.

[35] Zenon Chaczko Venkatesh Mahadevan, Shahrzad Aslanzadeh and Christopher Mcdermid, “Availability and Load Balancing in Cloud Computing”, 2011Page 70 International Conference on Computer and Software Modeling IPCSIT vol.14, IACSIT Press, Singapore, 2011.

[36] Zhang Bo; Gao Ji; Ai Jieqing “Cloud Loading Balance algorithm” Information Science and Engineering (ICISE), Second International Conference, 4-6 Dec. 2010.

[37] Architecture of cloud sim: https://en.wikipedia.org/wiki/CloudSim

[38] Specifications of the Cloud Analyst Workspace and Cloud Analyst tool: https://en.wikipedia.org/wiki/Cloud_analytics

[39] Definitions of load balancing in the cloud computing environment: https://en.wikipedia.org/wiki/Load_balancing_(computing)

[40] NIST definitions of the load balancing in cloud computing: https://csrc.nist.gov/publications/detail/sp/800-145/final

[41] Load Balancing in cloud computing definition of Foster: http://www.ianfoster.org/wordpress/2017/10/01/new-book-cloud-computing-for-science-and-engineering/